Apusic 金蝶天燕

生成时间：2026-05-19

开发指南

金蝶Apusic统一身份管理

版权声明

本文档所涉及的软件著作权、版权等知识产权已依法进行了注册，由金蝶天燕云计算股份有限公司合法拥有。受《中华人民共和国著作权法》《计算机软件保护条例》《知识产权保护条例》和相关国际版权条约、法律、法规以及其它知识产权法律和条约的保护。未经授权许可，不得非法使用。

免责声明

本文档包含的版权信息由金蝶天燕云计算股份有限公司合法拥有，受法律的保护，金蝶天燕云计算股份有限公司对本文档可能涉及到的非金蝶天燕云计算股份有限公司的信息不承担任何责任。在法律允许的范围内，您可以查阅并仅能够在《中华人民共和国著作权法》规定的合法范围内复制和打印本文档。任何单位和个人未经金蝶天燕云计算股份有限公司书面授权许可，不得使用、修改、再发布本文档的任何部分和内容，否则将被视为侵权，金蝶天燕云计算股份有限公司有依法追究其责任的权利。

本文档如有更新，不另行通知。对本文档中的问题您可向金蝶天燕云计算股份有限公司告知或查询。未经本公司明确授予的任何权利均予保留。

商标声明

Apusic是深圳市金蝶天燕云计算股份有限公司向中华人民共和国国家商标局申请注册的注册商标，注册商标专用权由金蝶天燕合法拥有，受法律保护。未经金蝶天燕的书面许可，任何单位及个人不得以任何方式或理由对该商标的任何部分进行使用、复制、修改、传播、抄录或与其它产品捆绑使用销售。凡侵犯金蝶天燕商标权的，金蝶天燕将依法追究其法律责任。本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

目录

1 概述
2 应用系统单点登录接入
- 2.1 协议说明
- 2.2 认证流程
- 2.3 第三方应用SDK接入单点步骤
  - 2.3.1 解压并引用SDK包
  - 2.3.2 创建oauth-config.properties文件
  - 2.3.3 应用系统到SSO认证
  - 2.3.4 应用系统获取用户信息
- 2.4 第三方应用API接入单点步骤
  - 2.4.1 登陆跳转
  - 2.4.2 第二步换取Access Token
  - 2.4.3 第三步换取User Info

3 用户名密码认证接口
- 3.1 获取token信息
- 3.2 解析token获取用户ID

4 第三方系统使用Restful接口
- 4.1 分页查询所有用户接口(无离职人员)
- 4.2 分页查询已授权访问某App的用户
- 4.3 查询所有机构接口
- 4.4 根据id查询机构接口
- 4.5 根据用户Id查询照片
- 4.6 根据用户id获取可以访问的app列表
- 4.7 查询用户在对应App中的权限列表

1 概述

统一身份认证平台上线后，将统一公司所有应用系统的账号管理和认证功能，统一身份认证平台将会为用户提供统一认证单点登录服务、统一自助服务、统一应用系统账号全生命周期管理服务；为达到以上功能，需要对接应用系统进行改造，改造内容如下图所示：

2 应用系统单点登录接入

SSO统一认证平台应用系统单点登录方式本次项目将统一采用OAuth标准协议接入所有应用系统

2.1 协议说明

OAuth（开放授权）是一个开放标准，允许用户让第三方应用访问该用户在某一网站上存储的私密的资源（如照片，视频，联系人列表），而无需将用户名和密码提供给第三方应用。

OAuth允许用户提供一个令牌，而不是用户名和密码来访问他们存放在特定服务提供者的数据。每一个令牌授权一个特定的应用在特定的时段内访问特定的资源。这样，OAuth允许用户授权第三方应用访问他们存储在另外的服务提供者上的信息，而不需要分享他们的访问许可或他们数据的所有内容。

2.2 认证流程

OAuth2.0认证流程步骤如下：

（一）用户 通过门户集成ParaSSO自服务平台访问第三方应用

（二）用户直接访问第三方应用

2.3 第三方应用SDK接入单点步骤

以下内容均为第三方应用需要做的工作，以J2EE应用平台为例说明：

2.3.1 解压并引用SDK包

1. 解压SDK 压缩包（apusic-oauth-sdk-1.0.rar），得到4个Jar包：commons-codec-1.3.jar 支持包

commons-logging.jar 支持包

jackson-all-1.9.1.jar 支持包

apusic-oauth-sdk-1.0.jar SDK包

2. 第三方应用工程引用4个Jar包，将4个Jar包拷贝至应用的WEB-INF/lib下或做外部引用（确保应用工程已经正常引用Jar包）

3. 备注：

A. 如果commons-codec-1.3.jar、commons-logging.jar、jackson-all-1.9.1.jar已经存在则无须再 引用

B. JDK版本支持JDK1.6以上

2.3.2 创建oauth-config.properties文件

1. 在自己的项目resources文件夹中创建oauth-config.properties配置文件（注意配置文件的名称不要变更）

2. 修改内容为等号后面部分

   ######################### Local OAuth Provider configuration ##########################

   config.base.url=http://lcy.aidm.apusic.com:9080/api/apusic/

   config.authorize.url=http://lcy.aidm.apusic.com:9080/api/apusic/oauth2/authorize

   config.accesstoken.url=http://lcy.aidm.apusic.com:9080/api/apusic/oauth2/token

   config.api.users.userinfo=http://lcy.aidm.apusic.com:9080/api/apusic/v1.0/oauth2/userinfo

############################ Local OAuth Provider configuration ######################## ############################## authorization_code ############################## config.authorization_code.client_id=711983296695394304

config.authorization_code.client_secret=PHOXMTQwNDIwMjIxNDEwMDQwODM7J1

config.authorization_code.redirect_uri=http://127.0.0.1:8080/oauth_callback

config.authorization_code.granttype=authorization_code

############################## authorization_code ##############################

############################## others url #####################################

config.queryuserbypage.url=http://lcy.aidm.apusic.com:9080/api/apusic/v1.0/userinfo

config.queryorg.url=http://lcy.aidm.apusic.com:9080/api/apusic/v1.0/orgs/tree

config.queryorgbyid.url=http://lcy.aidm.apusic.com:9080/api/apusic/v1.0/orgs

config.getuserphoto.url=http://lcy.aidm.apusic.com:9080/api/apusic/v1.0/personal/getphoto

config.queryuserapp.url=http://lcy.aidm.apusic.com:9080/api/apusic/v1.0/permissions/app

############################## others url #####################################

2.3.3 应用系统到SSO认证

此示例是应用系统在判断用户没有登陆自身应用系统后，重定向到SSO认证的代码，应用系统根据自身业务需要改造。一般都是添加在拦截器里面。

示例代码：

```
/**
* session过期或者用户未认证，拦截器
*/
@RequestMapping("/oauth_authentication")
public void doOauthAuthentication(HttpServletResponse httpResponse)
 throws IOException {
//配置信息应用系统存在自己的配置文件或者数据库里面，这里是举例子，存在配置文件里面
OAuthConfigUtil configUtil = new OAuthConfigUtil("oauthConfig");
OAuth20Config configInfo =
new OAuth20Config(configUtil.getClientId(),configUtil.getClientSecret(),
 configUtil.getRedirectUri(), configUtil.getAuthorizeUrl(),
configUtil.getAccessTokenUrl());
 IOAuth20Service service = new OAuthServiceBuilder(configInfo).build20Service();
 // 生成认证跳转地址
 String redUrl = service.getAuthorizationUrl();
 // 跳转到认证中心,进行认证
 httpResponse.sendRedirect(redUrl);
 return;
}
```

备注：一般加入到拦截器里面，拦截所有未认证请求

2.3.4 应用系统获取用户信息

示例代码：callback拦截器中获取用户身份信息

```
/**
* 认证中心认证成功后的调用的应用回调地址
*/
@RequestMapping("/oauth_callback")
public void oauthCallback(@RequestParam("code")String code, HttpServletRequest request, HttpServletResponse response) throws ServletException {
//配置信息应用系统存在自己的配置文件或者数据库里面，这里是举例子，存在配置文件里面
OAuthConfigUtil configUtil = new OAuthConfigUtil("oauthConfig");
OAuth20Config configInfo =
new OAuth20Config(configUtil.getClientId(),configUtil.getClientSecret(),
 configUtil.getRedirectUri(), configUtil.getAuthorizeUrl(),
configUtil.getAccessTokenUrl());
 IOAuth20Service service = new OAuthServiceBuilder(configInfo).build20Service();
 // 生成认证跳转地址，认证请求第一步请求回调地址不会传递code信息，要应用系统请求认证地址，认证中心第二次调用回调地址会将认code信息带回来。
If(null=code)
{
 String redUrl = service.getAuthorizationUrl();
 // 跳转到认证中心,进行认证，获取code信息
 httpResponse.sendRedirect(redUrl);
return;
}
//应用已经发起过认证请求，code信息已经传递过来
 try {
//根据code信息使用sdk中的方法获取token信息 oauthUser为成员变量
 oauthUser = new UserInfo(accessToken);
//根据token信息使用SDK中的方法获取用户登录信息 oauthUser为成员变量
 oauthUser = new UserInfo(accessToken);
 UserInfo loginUser = oauthUser.requestUserInfo(configUtil.getUserInfoUrl());
 认证过程完成，得到用户信息，下来为应用自身的访问逻辑
 }
 }
```

备注：应用的回调地址在SDK中也是一个参数需要传递进去

2.4 第三方应用API接入单点步骤

2.4.1 登陆跳转

用户登陆认证，访问应用域名或者认证平台域名（应用判断用户没有登录，将用户跳转到认证平台进行登录认证）：通过访问或者平台直接请求应用提供的回调地址,应用在接受到请求，发起到认证平台的认证请求（由应用服务器发起的用户客户端浏览器重定向）

```
http://lcy.aidm.apusic.com:9080/api/apusic/oauth2/authorize?
client_id=YOUR_CLIENT_ID&redirect_uri=YOUR_REGISTERED_REDIRECT_URI&response_type=code
```

示例请求url：

http://lcy.aidm.apusic.com:9080/api/apusic/oauth2/authorize?client_id=7119832966953943O4&redirect_uri=http://127.0.0.1:8080/oauth_callback&response_type=code

参数说明：

字段名	描述	备注
client_id	应用唯一标识，在应用注册的时候获得	在认证中心进行业务系统注册时，由认证中心进行分配
redirect_uri	URL ENCODE，应用callback回调URL	由业务系统开发并提供到认证中心进行注册，该callback回调地址需能接受code参数及target_uri参数
response_type	默认值，code	

用户在认证平台完成登录后，认证中心生成Code，并将用户重定向到应用系统的CallBack地址（该CallBack接收传递的Code参数）

示例连接地址：

http://127.0.0.1:8080/oauth_callback?code=586za9lGK7aK-hRgZrht-13pfDLmIzMWA1zhDfiilSbtDuwIfFebLiuTKw0M5dnHmW7swmGTVsKBQ93aaNPetGAe53ramnLWZ1vmSKJQes_LqzxNPo88wcN5DIBfU9Q1----返回给用户浏览器的callback地址，并带上code参数

2.4.2 第二步换取Access Token

应用得到Code，拿Code调用认证中心API换取Access Token，这个请求是非客户端浏览器请求，是由应用服务器直接发的http协议的请求(POST请求)

```
http://lcy.aidm.apusic.com:9080/api/apusic/oauth2/token?
client_id=YOUR_CLIENT_ID&client_secret=YOUR_SECRET&redirect_uri=YOUR_REGISTERED_REDIRECT_URI&code=CODE
```

示例请求url：

http://lcy.aidm.apusic.com:9080/api/apusic/oauth2/token?client_id=R9ZyHNCNbZ&client_secret=sdfjsdjfsfhsdhfsdhhfodsfsdf&redirect_uri=http%3A%2F%2Flocaliferay.cn.com%3A9082%2FOauthDome%2FLoginCode&code =34fdgdfgdfgf

参数说明：

字段名	描述	备注
client_id	应用唯一标识，在应用注册的时候获得	在认证中心进行业务系统注册时，由认证中心进行分配
client_secret	应用密钥，在应用注册的时候获得	在认证中心进行业务系统注册时，由认证中心进行分配
redirect_uri	URL ENCODE，应用callback回调URL,需要http转义	由业务系统开发并提供到认证中心进行注册，该callback回调地址需能接受code参数
code	用户本次访问的code	从上一步callback中得到

返回值说明：

示例参数：

access_token=623239c3760c600d81ed551e6d2adf4fwefwqd5a1ed24d8f959ac44b2b82b

字段名	描述	备注
access_token	接口调用凭证	在下一步换取用户信息的时候用到
expires	系统内部判断时用到，无需理会	

2.4.3 第三步换取User Info

应用得到Access Token后，拿Access Token调用认证中心API换取用户信息，这个请求是非客户端浏览器请求，是由应用服务器直接发的http协议的请求

```
http://lcy.aidm.apusic.com:9080/api/apusic/v1.0/oauth2/userinfo?access_token=ACCESS_TOKEN
```

示例请求url：

```
http://lcy.aidm.apusic.com:9080/api/apusic/v1.0/oauth2/userinfo?
access_token=R9ZyHNCNbZdsfsfsdfsfdsfsdfsdfsd
```

参数说明：

字段名	描述	备注
access_token	接口调用凭证	使用OAuth Code换取得到

返回值说明：

情况1：返回成功
`{"id":"10138640","attributes":[{"IDPEmail":["10138640"]}]}`
说明：``id为用户应用系统账号，attributes为其他属性信息

返回数据示例：

情况2：换取用户信息出错
{ "error":"expired_accessToken"}
说明：``换取信息出错，Access Token过期
情况3：sso中用户未与应用系统绑定账号
无返回值
说明：``此情况获取不到任何信息

3 用户名密码认证接口

此方法为用户在应用系统自身的登陆界面中，输入用户名/密码提交到服务端后，服务端通过接口向SSO发起用户认证校验的请求，用户名和密码校验通过后，会返回给应用系统用户身份信息给应用系统；

3.1 获取token信息

应用系统通过统一身份认证系统提供的接口，获取到用户身份信息，身份信息为token格式

示例url

http://144.131.246.210/profile/oidc/token?grant_type=password&scope=api &client_id=uHb6vpoZmj&client_secret=b231d8fa-4175-47a0-aa4e-8355fd344ba7&username=testCoremail2&password=xgNOk6$u

字段名	描述	备注
client_id	应用唯一标识，在应用注册的时候获得	在认证中心进行业务系统注册时，由认证中心进行分配
client_secret	应用密钥，在应用注册的时候获得	在认证中心进行业务系统注册时，由认证中心进行分配
username	用户登录sso使用的账号	
password	用户登录sso使用的密码	密码通过urlencode转码

3.2 解析token获取用户ID

应用系统获取到token信息为json格式字符串，需要去获取到id_token对应的值，然后用.进行分割，获得越高id_token的数组，取下标为1的值，然后base64解密，会获取到用户信息的json字符串，sub的值为用户对应的账号信息，应用系统获取到账号信息后，完成自己系统登录认证即可：

注，如果用户存在多个账号，sub上多个账号以逗号分割

单账号示例：

单账号加密字符串：

eyJleHAiOjE1NDU2NDY1ODEsInN1YiI6InRlc3RDb3JlbWFpbDEyIiwiYXRfaGFzaCI6Im85N2VJZXlCLV83RHJfcjY4TUlSVnciLCJhdWQiOlsia3pPZHBvSENUYiJdLCJpc3MiOiJ0ZXN0Q29y

单账号解密字符串：{"exp":1545646581,"sub":"testCoremail12","at_hash":"o97eleyB-_7Dr_r68MIRVw","aud":["kzOdpoHCTb"],"iss":"testCoremail12","iat":1545645981}

拥有的账号为：testCoremail12这个账号

多账号示例

多账号加密字符串：

eyJleHAiOjE1NzgzNjc0NzQsInN1YiI6IjEyMzQsdGVzdGVyLHRlc3QxIiwiYXRfaGFzaCI6IkFZNnZwTzVQQ19MRHQ2MFFaOWZNdGciLCJhdWQiOlsiSVZTaWs3TGZzYyJdLCJpc3MiOiI

多账号解密字符串：

{"exp":1578367474,"sub":"1234,tester,test1","at_hash":"AY6vpO5PC_LDt60QZ9fMtg","aud":["IVSik7Lfsc"],"iss":"1234,tester,test1","iat":1578366874}

拥有的账号为：1234、tester、test1这三个账号

应用系统获取到token信息为json格式字符串，需要去获取到id_token对应的值，然后用.进行分割，获得越高id_token的数组，取下标为1的值，然后base64解密，会获取到用户信息的json字符串，sub的值为用户对应的账号信息，应用系统获取到账号信息后，完成自己系统登录认证即可：

注，如果用户存在多个账号，sub上多个账号以逗号分割

单账号示例：

单账号加密字符串：

eyJleHAiOjE1NDU2NDY1ODEsInN1YiI6InRlc3RDb3JlbWFpbDEyIiwiYXRfaGFzaCI6Im85N2VJZXlCLV83RHJfcjY4TUlSVnciLCJhdWQiOlsia3pPZHBvSENUYiJdLCJpc3MiOiJ0ZXN0Q29y

单账号解密字符串：{"exp":1545646581,"sub":"testCoremail12","at_hash":"o97eleyB-_7Dr_r68MIRVw","aud":["kzOdpoHCTb"],"iss":"testCoremail12","iat":1545645981}

拥有的账号为：testCoremail12这个账号

多账号示例

多账号加密字符串：

eyJleHAiOjE1NzgzNjc0NzQsInN1YiI6IjEyMzQsdGVzdGVyLHRlc3QxIiwiYXRfaGFzaCI6IkFZNnZwTzVQQ19MRHQ2MFFaOWZNdGciLCJhdWQiOlsiSVZTaWs3TGZzYyJdLCJpc3MiOiI:

多账号解密字符串：

{"exp":1578367474,"sub":"1234,tester,test1","at_hash":"AY6vpO5PC_LDt60QZ9fMtg","aud":["IVSik7Lfsc"],"iss":"1234,tester,test1","iat":1578366874}

拥有的账号为：1234、tester、test1这三个账号

4 第三方系统使用Restful接口

4.1 分页查询所有用户接口(无离职人员)

注：查询用户列表(无离职人员，查询所有用户)

示例代码	@RequestMapping(value = "/queryuserbypage", method = RequestMethod.GET) public String queryUserByPage () { OAuthConfigUtil configUtil = new OAuthConfigUtil("oauthConfig"); UserInfoService service = new UserInfoServiceImpl(oauthUser.getAccessToken(), configUtil); PaginationUtil pagination = new PaginationUtil(); pagination.setPageNum(1); pagination.setPageSize(2); UserInfoQueryFilter filter = new UserInfoQueryFilter(); // filter.setExpireStatus(0); return service.queryUserByPage(pagination,filter); }
返回值	JSON串 { "data": [{ "authnType": 1, "badPasswordCount": 2, "badPasswordTime": "2021-11-02 09:56:34", "birthDate": "1997-01-01", "createBy": "admin", "createTime": "2020-10-10 10:00:00", "displayName": "张三", "email": "89465156@qq.com", "emailVerified": 1, "employeeNumber": 10051, "expireStatus": 0, "expireTime": "2029-09-09 10:00:00", "extendDataList": [], "familyName": "张", "gender": 1, "homeEmail": "15605469874@XX.com", "id": 622227747934179401, "idCardNo": 420116199901011234, "idType": 1, "isLocked": 1, "lastLoginIp": "", "lastLoginTime": "2021-11-02 09:56:34", "lastLogoffTime": "2021-11-02 09:56:34", "locale": "湖北省武汉市江汉区电力社区", "loginCount": 132, "managerType": 1, "married": 1, "mobile": 15607164562, "online": 0, "orgList": [], "passwordLastSetTime": "2021-11-02 09:56:34", "passwordSetType": 1, "picture": "", "quitDate": "2021-12-02", "rolesList": [], "startWorkDate": "2021-11-02 09:56:34", "status": 1, "tenantId": 0, "unLockTime": "2021-11-02 09:56:34", "updateBy": "admin", "updateTime": "2020-10-10 10:00:00", "userState": "RESIDENT", "userType": "EMPLOYEE", "username": "admin" }], "message": "请求成功", "pageInfo": { "current": 2, "pages": 4, "size": 20, "total": 62 }, "status": 0, "timestamp": 0 }

返回结果中参数

字段名	解释	数据类型
status	请求状态码，status为0表示请求成功，非0表示请求失败	
message	请求结果描述	
timestamp	时间戳	
data	请求数据	数组
pageInfo	分页信息	JSON对象
current	当前页数	integer
pages	总页数	integer
size	分页大小	integer
total	总条数	Integer
authnType	认证类型(用户名密码/手机)	integer
badPasswordCount	输入密码错误次数	integer
badPasswordTime	输入密码错误时间	String
birthDate	生日	String
createBy	创建者	String
createTime	创建时间	String
displayName	用户中文名称	String
email	邮箱	String
emailVerified	邮箱验证	integer
employeeNumber	员工编号	String
expireStatus	是否过期	String
expireTime	过期时间	String
extendDataList	用户扩展字段集合	List
familyName	姓	String

gender	性别（0未知;1女性;2男性）	integer
homeEmail	住址邮编	String
id	用户主键	String
idCardNo	身份证号	String
idType	证件类型(0未知;1身份证;2护照;3学生证;4军人证)	integer
isLocked	用户是否锁定	integer
lastLoginIp	用户末次登陆IP	String
lastLoginTime	用户末次登陆时间	String
lastLogoffTime	用户末次登出时间	String
locale	地点	String
loginCount	登录次数	integer
managerType	管理员类型(1是0否)	integer
married	婚姻状态（0未知 ;1单身;2结婚;3离异;4丧偶）	integer
mobile	手机号	String
online	是否在线(1在线0下线)	integer
orgList	用户所属组织集合	List
passwordLastSetTime	密码过期时间	String
passwordSetType	密码设置类型	integer
picture	照片	String
quitDate	离职日期	String
rolesList	用户所属角色集合	List
startWorkDate	开始工作时间	String
status	状态	integer
tenantId	租户	String
unLockTime	用户解锁时间	String
updateBy	修改人	String
updateTime	修改时间	String
userState	用户地区	String
userType	用户类型	String
username	用户名	String

4.2 分页查询已授权访问某App的用户

\1. 注：只能查询出应用账号的用户

示例代码	@RequestMapping(value = "/userapp",method = RequestMethod.GET) public String userApp() { OAuthConfigUtil configUtil = new OAuthConfigUtil("oauthConfig"); UserInfoService service = new UserInfoServiceImpl(oauthUser.getAccessToken(),configUtil); PaginationUtil pagination = new PaginationUtil(); pagination.setPageSize(2); pagination.setPageNum(1); //service.queryAppUserByPage(pagination,appId) return service.queryAppUserByPage(pagination,"711983296695394304"); }
返回值	Json串 { "status":0, "message":"请求成功", "timestamp":1650443612663, "data":{ "records":[{"createBy":null, "createTime":"2022-04-15 19:12:17", "updateBy":null, "updateTime":null, "tenantId":null, "id":null, "appId":"711983296695394304", "permissionsId":"683355466658615296", "permissionsName":"组一啊", "resourceId":"711983684903002112", "resourceName":"sdk_1", "resourceType":"APP", "description":null, "type":"GROUP", "status":1, "isPermit":1, "parentId":null,

"parentName":null }], "total":14, "size":2, "current":1, "orders":[], "optimizeCountSql":true, "hitCount":false, "countId":null, "maxLimit":null, "searchCount":true, "pages":7}}

返回结果中参数

字段名	解释
status	请求状态码，status为0表示请求成功，非0表示请求失败
message	请求结果描述
timestamp	时间戳
data	请求数据
total	数据总条数
size	每页包含条数
current	当前页码
orders	
optimizeCountSql	
hitCount	
countId	
maxLimit	
searchCount	
pages	总页数
records	用户数据
createBy	用户创建者
createTime	创建时间
updateBy	用户修改者
updateTime	修改时间
tenantId	租户Id
id	用户Id
appId	AppId
permissionsId	被授权的Id
permissionName	授权对象名
resourceId	资源Id
resourceName	资源名称
resourceType	资源类型
description	资源描述
type	资源类型
status	资源状态
isPermit	是否允许使用
parentId	父节点Id
parentName	父节点名称

4.3 查询所有机构接口

示例代码	@RequestMapping(value = "/queryorg", method = RequestMethod.GET) public String queryOrg () { OAuthConfigUtil configUtil = new OAuthConfigUtil("oauthConfig"); OrgService service = new OrgServiceImpl(oauthUser.getAccessToken(),configUtil); return service.queryOrg(); }
返回值	Json串 { status: 0, message: "请求成功", timestamp: 1650443109687 data: [children: null description: null disabled: false extendId: null id: "0" level: 0 name: "Root" namePath: null parentId: "-1" sortIndex: 0 tenantId: "" type: null}] }

返回结果中参数

字段名	解释
status	请求状态码，status为0表示请求成功，非0表示请求失败
message	请求结果描述
timestamp	时间戳
data	请求数据
id	组织主键
name	组织名称
type	组织类型
level	当前层级
parentId	父节点Id
description	组织描述
tenantId	租户Id
sortIndex	排序编号
children	子组织集合
disabled	该组织是否可选，如果组织已经被选定，则disabled为true，不能再进行选择添加
namePath	名称路径
extendId	外部编号

4.4 根据id查询机构接口

示例代码	@RequestMapping(value = "/queryorgbyid", method = RequestMethod.GET) public String queryOryById() { OAuthConfigUtil configUtil = new OAuthConfigUtil(" OrgService service = new OrgServiceImpl(oauthUser.getAccessToken(),configUtil); // service.queryOrgById(orgId) return service.queryOrgById("1a23a7de97634050b1 }
返回值	Json串 { "status":0, "message":"请求成功", "timestamp":1650442461926, "data":{ "id":"1a23a7de97634050b1ee66ef06300407", "name":"测试日志1", "type":null, "level":1, "p "description":null, "tenantId":null, "sortIndex":17, "children":[{"id":"9f9ed137a15844b293dc766e6e3ab416","name":"1111","type":null,"level":2,"parentId":"1a23a7de97634050b1ee66ef06300407","description":null,"tenantId":null,"sortI [{"id":"ca99b4bec97f4c4b8640954fe474759e","name":"2222","type":null,"level":3,"parentId":"9f9ed137a15844b293dc766e6e3ab416","description":null,"tenantId":null,"sor [],"extendId":null,"namePath":"测试日志1/1111/2222","disabled":false}],"extendId":null,"namePath":"测试日志1/1111","disabled":false}],"extendId":null,"namePath":"测试日志

返回结果中参数

字段名	解释
status	请求状态码，status为0表示请求成功，非0表示请求失败
message	请求结果描述
timestamp	时间戳
data	请求数据
id	组织主键
name	组织名称

type	组织类型
level	当前层级
parentId	父节点Id
description	组织描述
tenantId	租户Id
sortIndex	排序编号
children	子组织集合
disabled	该组织是否可选，如果组织已经被选定，则disabled为true，不能再进行选择添加
namePath	名称路径
extendId	外部编号

4.5 根据用户Id查询照片

示例代码	@RequestMapping(value = "/getuserphoto", method = RequestMethod.GET) public String getUserPhoto() { OAuthConfigUtil configUtil = new OAuthConfigUtil("oauthConfig"); UserInfoService service = new UserInfoServiceImpl(oauthUser.getAccessToken(),configUtil); //service.getUserPhoto(userId) return service.getUserPhoto("1"); }
返回值	JSON串 { "status":0, "message":"请求成功", "timestamp":1650442083184, "data": { "picture":"/minio/aidm1/2022/3/30/aidm_06387344a1b24b0ba82faf08ac1233cf.png" } }

返回结果中参数

字段名	解释
status	请求状态码，status为0表示请求成功，非0表示请求失败
message	请求结果描述
timestamp	时间戳
data	请求数据
picture	用户头像url

4.6 根据用户id获取可以访问的app列表

示例代码	@RequestMapping(value = "/apps", method = RequestMethod.GET) public String apps() { OAuthConfigUtil configUtil = new OAuthConfigUtil("oauthConfig"); UserInfoService service = new UserInfoServiceImpl(oauthUser.getAccessToken(),configUtil); PaginationUtil pagination = new PaginationUtil(); pagination.setPageSize(1000); pagination.setPageNum(1); // service.getAppsByUserId(pagination,userId) return service.getAppsByUserId(pagination,"712381959854510080"); }
返回值	JSON串 { "status":0, "message":"请求成功", "timestamp":1650533129399, "data":[{ "id":"709119375139016704", "authDomainName":null, "appId":"709119311897300992", "name":"授权组织", "status":1, "resourceType":"APP", "resourceUrl":null, "createTime":"2022-04-06 16:28:19", "resourceIcon":"/minio/aidm1/2022/4/6/aidm_afa5175d06204403ab4c83cc628d2d3a.png", "subAccountStatus":0, "protocol":"OAuth_v2.0", "childPasswordStatus":0 }] }

返回结果中参数

字段名	解释
status	请求状态码，status为0表示请求成功，非0表示请求失败
message	请求结果描述
timestamp	时间戳
data	请求数据

id	资源id
authDomainName	权限分组名称
appId	应用id
name	资源名称
status	资源状态(0表示用户自己的资源可以删除，1表示分组和角色的资源不可删除)
resourceType	资源类型
resourceUrl	资源标识
createTime	创建时间
resourceIcon	资源图标
subAccountStatus	是否拥有子账户(1表示有子账户 0表示没有子账户)
protocol	单点登录类型
childPasswordStatus	是否需要子账号密码

4.7 查询用户在对应App中的权限列表

示例代码	@RequestMapping(value = "/permissionsapps", method = RequestMethod.GET) public String permissionsApps() { OAuthConfigUtil configUtil = new OAuthConfigUtil("oauthConfig"); UserInfoService service = new UserInfoServiceImpl(oauthUser.getAccessToken(),configUtil); // service.getPermissionsUser(appId,userId) return service.getPermissionsUser("1","1"); }
返回值	JSON串 { "status":0, "message":"请求成功", "timestamp":1651025616837, "data":[{ "resourceId":"aidm:menu:home", "resourceName":"首页", "resourceKey":null }] }

返回结果中参数

字段名	解释
status	请求状态码，status为0表示请求成功，非0表示请求失败
message	请求结果描述
timestamp	时间戳
data	请求数据
resourceId	资源Id
resourceName	资源名称
resourceKey	第三方应用在录入时设定的第三方应用的资源在对应应用中的权限标识

全国统一服务热线

4008-555-800

金蝶天燕云计算股份有限公司(简称“金蝶天燕云”)成立于2000年，前身为“金蝶中间件公司”，是金蝶集团旗下新一代软件基础云平台服务商，云计算国家标准制定企业，国家信创产业核心软件企业。金蝶天燕是国家863重点研发计划与核高基重大专项承接企业，也是“两网一站四库十二金”国家重点工程的基础平台提供商，产品广泛应用于政府、军工、金融、能源等关键行业，累计服务客户总数超过10万家。

Apusic 金蝶天燕

云计算国家标准制定企业

金蝶集团旗下基础软件企业

信息技术应用创新核心企业

官网：www.apusic.com