iuap 红皮书
友户通开发

2022 年 5 月

版权

本文摘要

友户通建立了用友 BIP 产品的用户、企业账号（租户）、应用的统一业务模型，为上层云服务提供了用户管理、单点登录、企业账号（租户）管理、应用开通和访问控制功能。

本文主要介绍了友户通提供的基础管理功能、集成开发方式和典型场景的客开方案。通过对本文的学习，可以了解 BIP 产品的登录验证管理、用户管理，企业账号（租户）管理的使用方式，以及友户通对外提供的集成能力，方便项目实施人员根据客户实际场景合理配置系统和二次开发。

基础管理功能在第二章，包括三个部分：

●系统基础配置

包括登录验证类手机短信、邮箱验证码配置（业务通知类短信、邮箱配置请参考消息开发红皮书）；密码复杂度、有效期配置；用户自行修改个人信息范围限制等。

●企业账号（租户）管理

包括企业账号（租户）的创建、初始化。

●用户运营

提供管理系统所有用户功能，包括密码修改、重置，基本信息修改，查询用户所属企业账号，注销等。只有系统超级管理员 yhtmanager 可以使用这个节点。

集成开发方式和典型场景的客开方案部分在第三、四、五、六章，其中第三章应用集成友户通是后续章节的基础，包括以下部分：

●客开系统接入友户通 SDK 的方式（第三章）

●部分常用接口定义和客开系统接入友户通 CAS 单点登录方案（第四章）

●自定义短信通道对接方案（第五章）

●外部系统单点登录用友 BIP 集成方案（第六章）

包括专属化接口方案、专属化 AD 域集成方案、公有云集成认证中心方案。

文档修订摘要

日期	修订号	描述	著者	审阅者
2022-5-25	1	添加短信、单点	SFH	
2022-6-6	2	添加 AD 域集成	JSB	
2022-6-11	3	明确集成友户通配置项	SFH	
2022-6-15	4	更新图片，添加邮箱、短信配置常见问题	SFH、XZH	
2022-6-19	5	增加摘要	SFH	

目录

第一章 概述

友户通，是用友 BIP 的基础核心产品，建立了用户、企业、企业帐号、应用等核心实体的统一业务模型，为上层云服务提供了清晰的多租户架构、用户管理和应用开通标准化流程。

友户通，提供统一的用户管理和企业管理服务，可统一存储和管理所有用户和企业信息，为用友云及生态应用提供统一的用户和企业数据，并且提供统一身份管理服务，集中进行身份认证、授权和访问控制服务。

友户通专属云提供的登录方式多样，支持的身份授权标准丰富，并可依据企业需求定制密码校验策略，满足企业不同的安全等级要求。同时，友户通提供的联邦用户认证服务更可在不替换企业现有用户中心的基础上将友户通用户中心和企业现有用户中心进行集成，实现单点登录。

第二章 功能介绍

2.1 名词解释

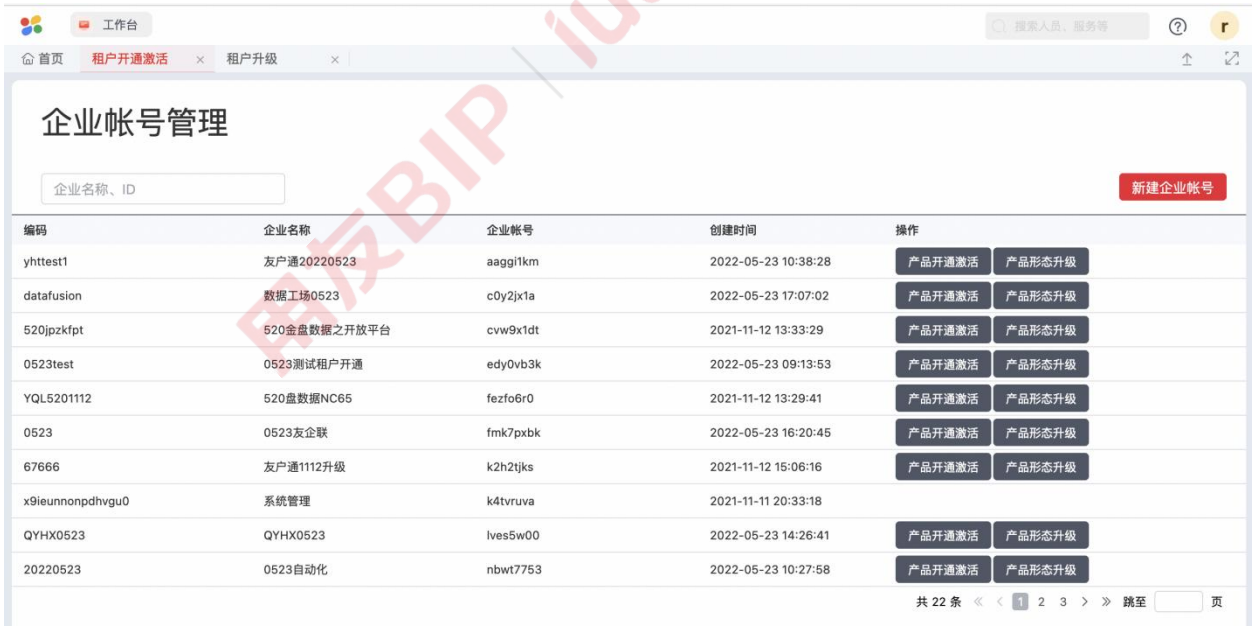
- 友户通：提供统一的用户管理、企业管理服务，可统一存储和管理所有用户和企业信息，为用友云及生态应用提供统一的用户和企业数据，并且提供统一身份管理服务，集中进行身份认证、授权和访问控制服务。
- 用户账号：简称用户，是云产品、软件产品的具体使用人员的账号。一个用户一个ID，终身唯一，是进入、通行用友云的“身份证”，是全局唯一的。同一个手机号码只能注册一个友户通账号，同一个邮箱只能注册一个友户通账号。
- 用户身份：用户身份是 YonBIP 为了支持多种产品场景而特殊设置的概念，用户身份也可以理解为一组角色的集合，在不同产品里可以支持不同类型的用户，比如普通员工，供应商，客户。
- 企业账号（租户）：是企业在友户通的账号，一个企业可以拥有多个企业账号，便于隔离、方便管理，可以理解为企业的多个子账号。企业账号是基础的业务控制数据信息，是数据隔离、业务隔离、流程隔离、资源隔离、计费隔离等的依据。
- 应用管理中心管理的产品：是用友云中可注册、开通的所有产品，既包含财务云、人力云等云服务产品，也包含 U8、U8 Cloud 等云注册产品。企业账号购买和激活应用后，再授权企业的部分用户使用和管理。
- 单点登录(SingleSignOn，SSO)：通过用户的一次性鉴别登录。当用户在身份认证服务器上登录一次以后，即可获得访问单点登录系统中其他关联系统和应用软件的权限，同时这种实现是不需要管理员对用户的登录状态或其他信息进行修改的，这意味着在多个应用系统中，用户只需一次登录就可以访问所有相互信任的应用系统

2.2 功能描述

2.2.1 创建企业账号

默认用户 yhtmanager 默认密码 manager@2020 登录工作台：

开通后在默认的企业账号下，可以选择“系统管理”企业账号，在数字化建模下的租户开通激活节点中新建企业帐号。



2.2.2 基础配置

使用 yhtmanager 账号登录系统，选择“系统管理”租户，然后在数字化建模下找到基础配置节点。

2.2.2.1 邮件配置

点击【邮件配置】。

Smtphost：是邮件 **SMTP** 协议服务器地址；smtpport：邮件 **SMTP** 协议端口；(开发者中心部署模式服务器地址需要配置成对应的 IP)

Imaphost：是邮件 **IMAP** 协议服务器地址；imapport：邮件 **IMAP** 协议端口；(开发者中心部署模式服务器地址需要配置成对应的 IP)

account:登录邮件服务器的用户名；password:登录邮件服务器的密码（**此密码为 AES 加密密文，加密地址：http://部署环境/yht-user/genpsw,需要登录后才能访问**）

【备注】：(配置保存成功,等待 10 分钟配置完成更新或重启服务更新配置)

常见问题

1.网络

ping mail.yonyou.com 实际使用时候替换成 smtpHost 配置。下图所示是网络正常。

```
[root@dciuap2 ~]# ping mail.yonyou.com
PING mail.yonyou.com (192.168.210.160) 56(84) bytes of data.
From 172.20.253.134 (172.20.253.134) icmp_seq=1 Packet filtered
From 172.20.253.134 (172.20.253.134) icmp_seq=2 Packet filtered
From 172.20.253.134 (172.20.253.134) icmp_seq=3 Packet filtered
From 172.20.253.134 (172.20.253.134) icmp_seq=4 Packet filtered
From 172.20.253.134 (172.20.253.134) icmp_seq=5 Packet filtered
```

下图是网络不通，需要联系现场运维检查网络问题。

```
root@test-iuap-uuas-user-75596f644c-9bmdv /app # ping mail.yonyou.com
PING mail.yonyou.com (192.168.210.160): 56 data bytes
```

2.邮箱配置

点击

常见异常

535 Error: authentication failed

账号密码错误或邮件服务账号密码使用授权码加密

Couldn't connect to host, port: 192.168.210.1601, 25; timeout -1

smtpHost/smtpPort 配置错误，请检查

Could not connect to SMTP host: 192.168.210.160, port: 25

smtpSsl 和 imapSsl 默认 false/不勾选

554 sender is rejected

退信原因：发信人地址被对方屏蔽，也就是被对方加入了黑名单拒收的地址中，如果不能发给对方域名下的多个联系人，则表明是全局黑名单，整个域名都禁止你的来信。

解决方案：换用其它域名的邮箱或者私人邮箱跟对方联系，或者电话沟通要求对方解封。

2.2.2.2 短信配置

点击【短信配置】。系统配置- 短信配置支持 APILink 无模板发送通道，使用前需有 APILink 申请授权。（Apilink 官方地址 https://api.yonyoucloud.com/apilink，联系官网客户电话 010-86393388）。

系统配置-短信配置中，主要需要配置 sendNoTemplateUrl 和 noTemplateApicode 两项，发送国际短信需要配置 internationalApicode，其余配置项若无需修改使用默认值即可。如果不使用 ApiLink 短信服务，可将短信发送的类文件放在网关类下，com.yonyou.yht.gateway.message.message1 路径下，短信网关会自动加载该路径下的 MessageSender1。如需添加备用通道可在 com.yonyou.yht.gateway.message.message2 下放入 MessageSender2。其中 MessageSender1 和 MessageSender2 必须实现以下 send 方法：

Publicboolean send(String mobile,Stringmsg, String[] codes, String templateCode, String countryCode, String apicode)

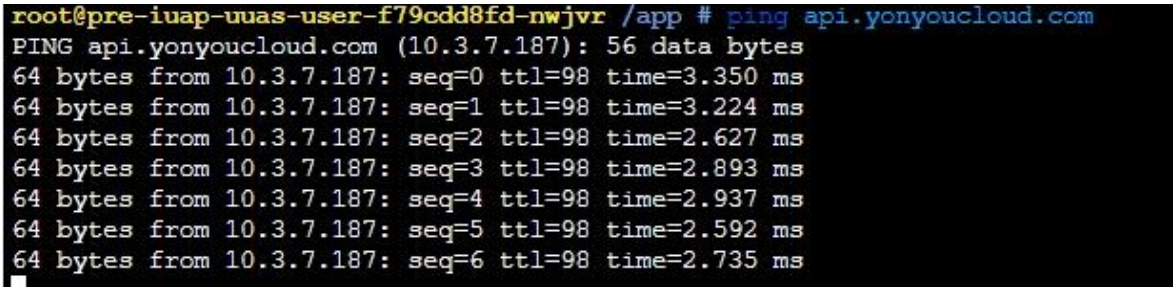
【备注】(配置保存成功,等待 10 分钟配置完成更新或重启服务更新配置)

如果需要使用客户自定义短信通道，需要客开适配器，具体参见 5.1 章节。

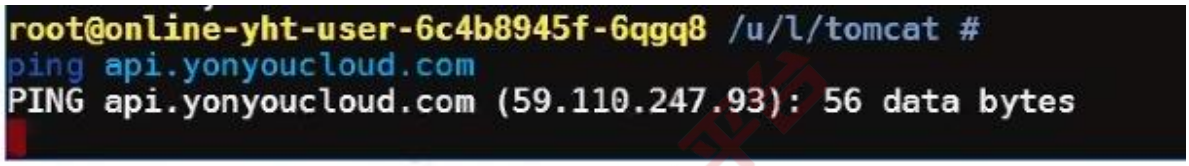
常见问题

1. 网络

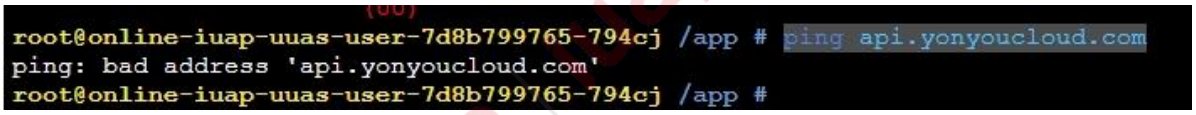
ping api.yonyoucloud.com 实际使用时候替换成 sendNoTemplateUrl 配置中的域名。下图所示是网络正常。



下图是网络不通，需要联系现场运维检查网络问题。



下图是 DNS 解析失败，需要联系现场运维检查 DNS 问题。



2.2.2.3 登录过期时间和注销前缀

点击【过期时间和注销前缀】。

用户帐号前缀：设置注销用户的前缀标识，注销后的用户会被限制登录。【备注】(要求中文字符《开头，数字和字母组合)

2.2.2.4 兑换 Token 接口的 IP 白名单

点击【Token 接口的 IP 白名单】。可以输入多个 IP 地址进入白名单。

2.2.2.5 登录白名单

点击【登录白名单】。可以输入多个 IP 地址进入白名单。

是否开启登录白名单，开启后登录的白名单生效。

2.2.2.6 属性修改控制

点击【属性修改控制】，能够对企业级的用户名、手机号、邮箱、账号、密码控制是否可以修改。

2.2.2.7 验证码模板配置

点击【验证码模板】。

系统配置-验证码模板配置是指对发送的短信内容和邮件内容的配置，二者使用同一模板，在输入框中输入自定义的短信模板。企业标签配置需和 APILink 官网地址申请注册的短信标签保持一致。

【备注】自定义短信通道需要自己修改短信签名。修改【企业标签(中文)】和【企业标签(英文)】配置对应 APILink 官网地址申请注册的短信标签，短信模板配置生效后接收短信如下图所示：

2.2.2.8 密码策略配置

点击【密码策略配置】。

默认密码：manager@2020，企业服务中心导入用户的默认密码。

出错次数：用户输入密码错误次数超过出错次数时帐号将被锁定，锁定时间结

束自动解锁或者让用户找回密码解锁。

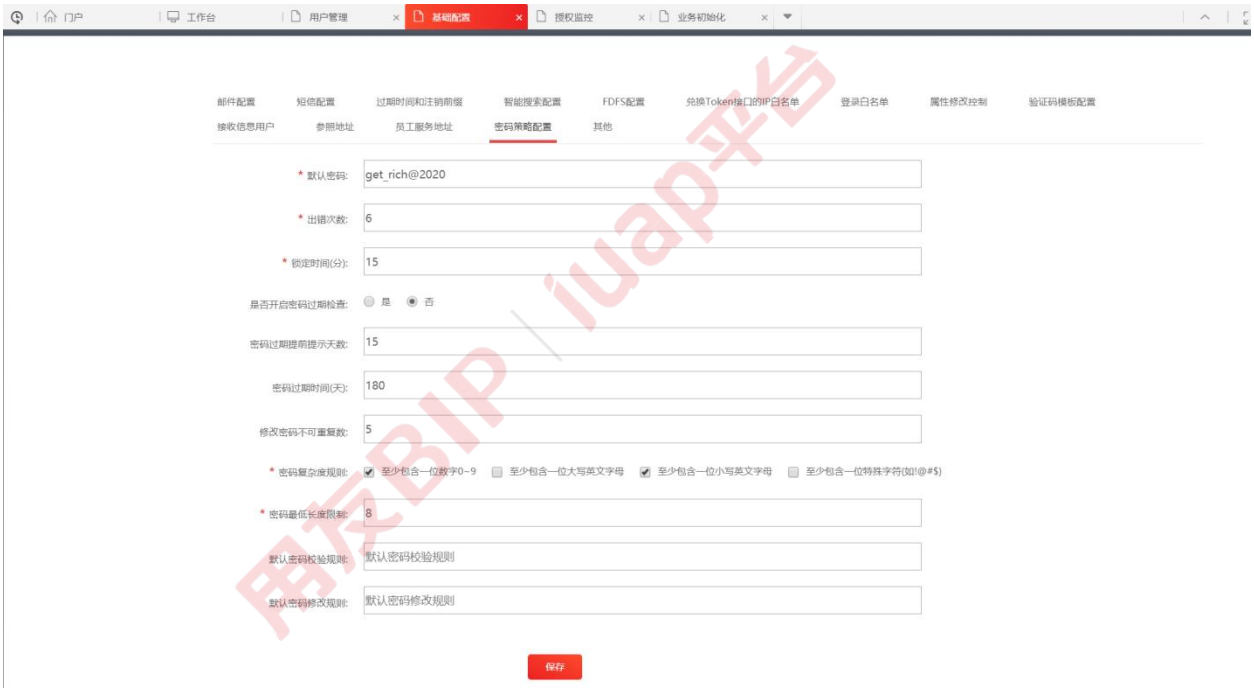
锁定时间：默认锁定时间 900 秒（15 分钟），单位秒。

是否开启密码过期检查：默认关闭，如果开启用户长时间未修改密码登录时会提醒修改密码。

密码过期时间：用户密码的过期时间，默认 180 天，单位天。

密码过期提前提示天数：密码过期提前提示的时间，单位天。密码快过期时用户登录即提示修改密码。

修改密码不可重复数：用户修改密码不可与之前修改的密码重复的次数。



2.2.2.9 其他

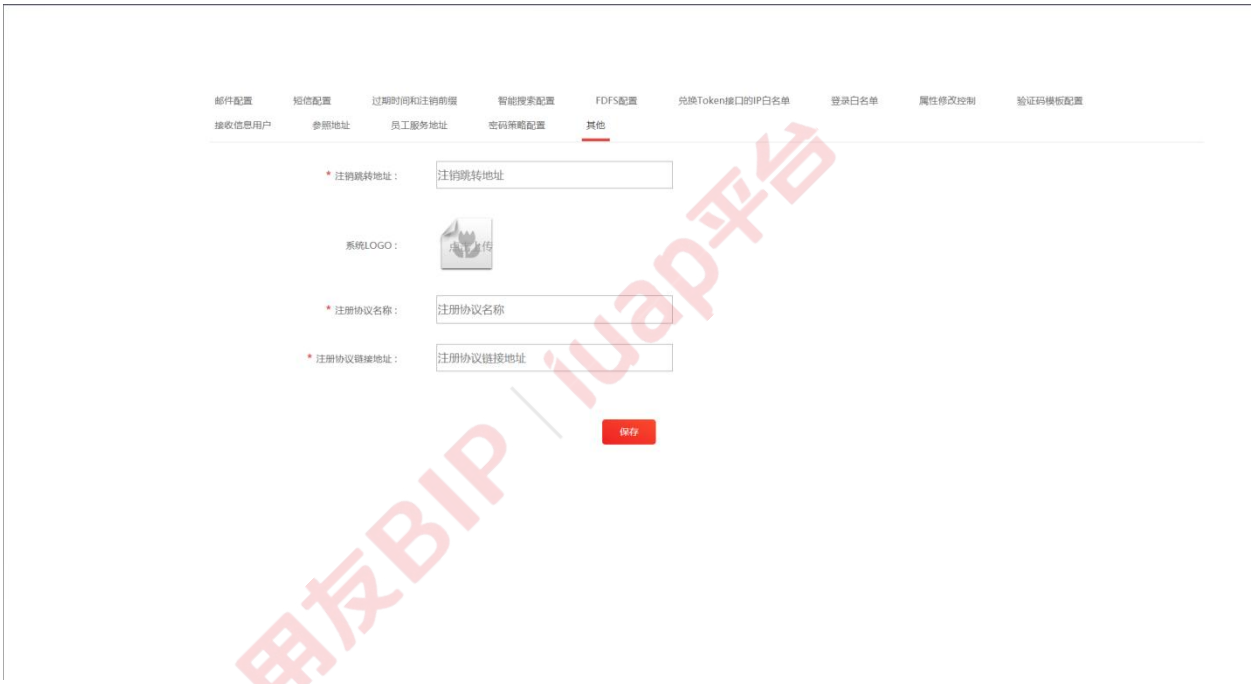
点击【其他】。

注销跳转地址：配置用户中心，控制台服务注销后跳转页面的地址；

系统 LOGO：配置用户中心、注册页面、找回密码，控制台等左上角 LOGO 图片，图片高度约 80px，宽度不限，建议底色透明。

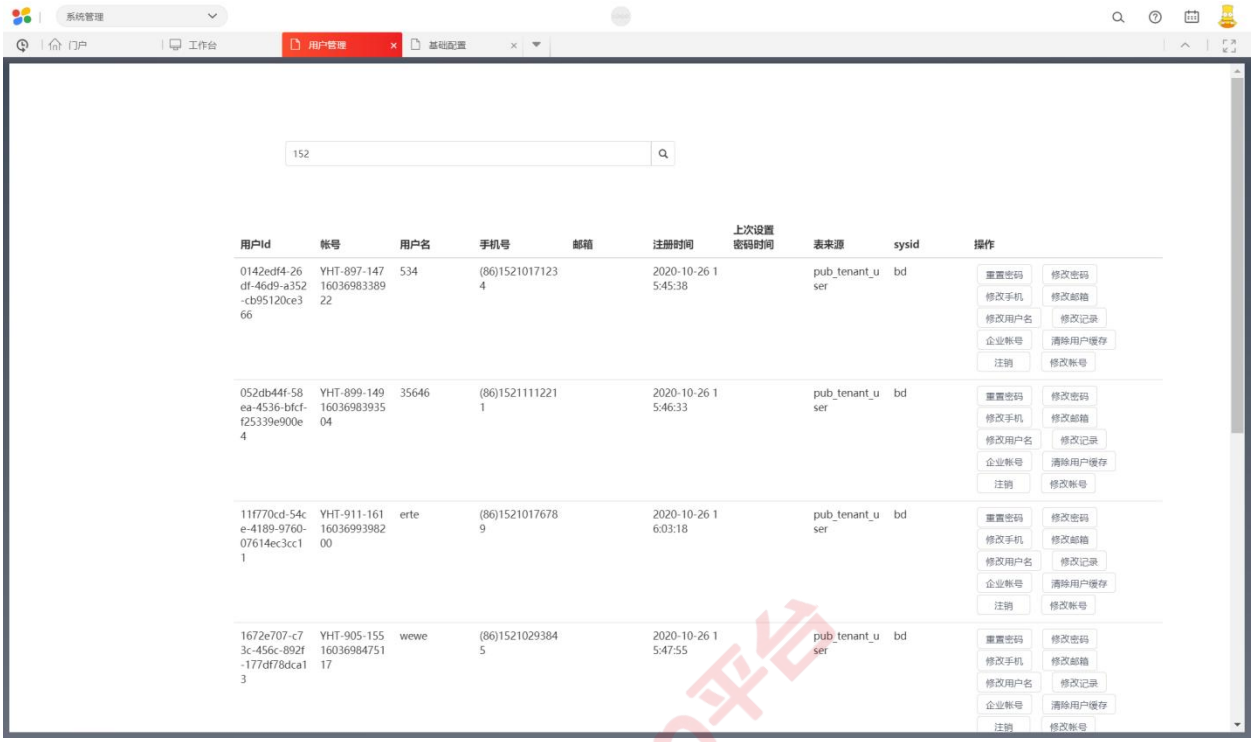
注册协议名称，注册协议链接地址：配置用户中心注册页面中的注册协议名称，注册协议链接地址，配置生效后效果如下图所示：

完成输入后点击保存，控制台页面刷新即可生效，用户中心、注册、找回密码页面配置效果 10 分钟后刷新页面生效。



2.2.3 用户运营

用户运营节点（之前叫用户管理）提供企业帐号管理员管理用户的功能，模糊查询可采用用户 ID\用户名\邮箱\手机号查询用户。



以下各按钮的功能描述

重置密码：用户密码重置为默认密码

修改密码：用户密码修改为指定的密码

修改手机：修改当前用户手机号信息

修改邮箱：修改当前用户邮箱信息

修改用户名：修改当前用户用户名信息

修改记录：修改操作的相关记录信息

企业帐号：用户关联的企业帐号列表

清除用户缓存：清除 redis 中当前用户信息

修改帐号：修改“帐号”属性信息

注销：销毁当前有户

第三章 应用集成友户通

3.1 获取授权文件配置

专属云统一采用默认应用 iuap5 的授权文件，从 YMS 控制台左侧菜单列表选择 YMS 中间件配置--》友户通全局配置，可获取。

YMS 中的配置项与 sdk.properties 以及 authfile 中配置项映射关系如下，基本 YMS 中的配置项比两个文件中的多了 yht.前缀。

文件名	文件中置项名称	YMS 中配置项名称
sdk.properties	apps.tenant.base.url	yht_apps_tenant_base_url
sdk.properties	yht.user.base.url	yht_user_base_url
sdk.properties	cas.url	yht_cas_url
sdk.properties	yht.enterprise.base.url	yht_enterprise_base_url

sdk.properties	servername	yht.servername
sdk.properties	sysid	yht.sysid
sdk.properties	secretKey	yht.secretKey
sdk.properties	httpclient.MaxTotal	yht.httpclient.MaxTotal
sdk.properties	httpclient.MaxPerRoute	yht.httpclient.MaxPerRoute
sdk.properties	yht.connect.timeout	yht.connect.timeout
sdk.properties	ignorePattern	yht.ignorePattern
authfile	appId	yht.appId
authfile	username	yht.username
authfile	key	yht.key
authfile	client_id	yht.client_id
authfile	client_secret	yht.client_secret
authfile	expiredTs	yht.expiredTs
authfile	scope	yht.scope

3.2 接入方式

3.2.1 配置 pom 文件

友户通的集成提供 SDK 的接入形式，以 maven 工程为例，在 pom 文件里添加依赖包，父 pom 需要指定为 iuap-2nd-party，具体版本号根据实际情况填写。

```
<<parent>
    <groupId>com.yonyou.iuap</groupId>
    <artifactId>iuap-2nd-party</artifactId>
    <version>具体版本号根据实际情况填写</version>
</parent>

<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>

<dependencies>
    <dependency>
        <groupId>com.yonyou.yht</groupId>
```

```
        <artifactId>yht-sso-client</artifactId>
    </dependency>
    <dependency>
        <groupId>com.yonyou.yht</groupId>
        <artifactId>yht-sso-cache</artifactId>
    </dependency>
    <dependency>
        <groupId>com.yonyou.yht</groupId>
        <artifactId>yht-client-common</artifactId>
    </dependency>
    <dependency>
        <groupId>com.yonyou.yht</groupId>
        <artifactId>yht-sdk</artifactId>
    </dependency>
    <dependency>
        <groupId>com.yonyou.iuap</groupId>
        <artifactId>iuap-tenant-cas</artifactId>
    </dependency>
    <dependency>
        <groupId>com.yonyou.iuap</groupId>
        <artifactId>iuap-tenant-sdk</artifactId>
    </dependency>
</dependencies>
```

3.2.2 配置 sdk.properties 文件

使用 YMS 管理配置项的应用不需要这个步骤。

在 classes 路径下新建 sdk.properties，配置内容如下，配置项的值参考 3.1 章节。

```
yht.user.base.url=
apps.tenant.base.url=
yht.enterprise.base.url=
cas.url=
servername=
sysid=
yht.client.credential.path=authfile.txt
oauth2file.path=authfile,txt
```

3.2.3 配置 authfile.txt 文件

使用 YMS 管理配置项的应用不需要这个步骤。

在 sdk.properties 相同路径下新建 authfile.txt，配置内容如下，配置项的值参考 3.1 章节。

```
expiredTs=
key=
appId=
username=
```

```
client_secret=
redirect_uri=http.*
scope=
client_id=
```

第四章 开发接入说明

使用如下接口需要先按照第三章集成友户通。

4.1 API 说明

4.1.1 根据用户 ID 获取用户信息

接口：com.yonyou.yht.sdk.UserCenter

接口类型：rest

接口说明：根据用户 ID 获取用户信息

1） getUserById

方法说明：根据用户 id，获取用户详细信息

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
userId	String	用户 ID	true	

响应参数：

参数名称	参数类型	参数说明	示例值
status	String	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
user	UserVO	用户基本信息	

请求示例：

```
String result = com.yonyou.yht.sdk.UserCenter.getUserById(userId);
SDKJsonResponse jr = SDKResultUtils.getData(result, "status", "msg", "user");
```

响应示例：

```
{
        "status":  "1",
```

```
"user":{
...
}
}
```

4.1.2 根据登录名获取用户信息

接口：com.yonyou.yht.sdk.UserCenter

接口类型：rest

接口说明：根据登录名获取用户信息

方法名：getUserByLoginName

方法说明：根据登录名，获取用户详细信息

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
userName	String	用户登录名	true	

响应参数：

参数名称	参数类型	参数说明	示例值
status	String	请求的状态，如果为 0，表示失败，如果为，表示成功	
user	UserVO	用户基本信息	

请求示例：

```
String result = com.yonyou.yht.sdk.UserCenter.getUserByLoginName (userName);
SDKJsonResponse jr = SDKResultUtils.getData(result, "status", "msg", "user");
```

响应示例：

```
{
"status": "1",
"user":{
...
```

```
}
}
```

4.1.3 根据登录名模糊查询用户列表

接口：com.yonyou.yht.sdk.UserCenter

接口类型：rest

接口说明：据登录名模糊查询用户列表（支持分页）

方法名：searchUserByName

方法说明：据登录名模糊查询用户列表

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
name,	String	用户名（可以是用户编码，用户邮箱或者手机号），必须是数字或字母开头	true	
pageSize	String	页面大小		
pageNumber	String	第几页		
sortType	String	排序值，默认值为“auto”,此时按“userCode”字段排序，当值为“name”时按“userName”字段排序，不支持其他字段的排序		

响应参数：

参数名称	参数类型	参数说明	示例值
status	String	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
users	List<User>	用户列表	

请求示例：

```
String result = com.yonyou.yht.sdk.UserCenter.searchUserByName (username,"10","1","desc");
SDKJsonResponse jr = SDKResultUtils.getData(result, "status", "msg", "users");
```

响应示例：

```
{
"status":  "1",
"users":[{
…// 用户信息
},{
}]
}
```

4.1.4 根据 userCode、userMobile、userEmail 判断用户是否已存在

接口类：com.yonyou.yht.sdk.UserCenter

接口类型：rest

请求方法：isUserExist

方法说明：根据 userCode、userMobile、userEmail 判断用户是否已存在

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
usercode	String	用户帐号（编码）	Fale	
userMobile	String	用户手机号（userMobile,userEmail 不能同时为空）	true	
userEmail	String	用户邮箱	true	

响应参数：

参数名称	参数类型	参数说明	示例值

status	int	如果为 0，表示手机号和邮箱不能同时为空； 如果为 1，表示用户帐号已经存在且手机号一致，用户已存在； 如果为 2，表示用户帐号已经存在且邮箱一致，用户已存在； 如果为 3，表示用户帐号已经存在且手机号,邮箱一致，用户已存在； 如果为 4，表示用户帐号已存在； 如果为 5，表示手机号已存在； 如果为 6，表示邮箱已存在； 如果为 7，表示用户帐号,手机号,邮箱都不存在，可以创建新用户。 如果为 8，表示用户手机号作为另一个用户的 usercode 已经存在。 如果为 9，表示用户邮箱作为另一个用户的 usercode 已经存在。 如果为 10，表示当 userEmail 已存在数据库中，且邮箱手机号匹配，不可以添加新用户,未检查 usercode	

msg	String	status 为 0:返回“参数有问题，手机号和邮箱不能同时为空”； status 为 1:返回“用户帐号已经存在且手机号一致，用户已存在”； status 为 2：返回“用户帐号已经存在且邮箱一致，用户已存在”； status 为 3:返回“用户帐号已经存在且手机号，邮箱一致，用户已存在”； status 为 4:返回“用户帐号已存在”； status 为 5:返回“手机号已存在”； status 为 6:返回“邮箱已存在”； status 为 7:返回“不存在相关用户，可以创建新用户”。 status 为 8:返回“该手机号作为另一个用户的 usercode 已经存在”。 status 为 9:返回“该邮箱作为另一个用户的 usercode 已经存在”。 status 为 10：返回“邮箱手机号已存在”。	
user	UserVO	用户基本信息；当 status 为 3、5、6 或 10 时有该字段，否则无该字段。	

响应示例：

```
{
  "status" : 7,
  "msg" : "不存在相关用户，可以创建新用户"
}
```

4.1.5 根据用户 id 查询所有的租户

接口：com.yonyou.iuap.tenant.sdk.TenantCenter

接口类型：rest

方法名：findTenantsByUserId

方法说明：根据用户 id 查询该用户在该产品可登录的租户。

如果租户管理员开启授权，该用户没有授权该产品，那么该租户不可登录。

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
userId	String	用户名 Id	true	

响应参数：

参数名称	参数类型	参数说明	示例值
status	String	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
tenants	List<Tenant>	租户列表	

请求示例：

```
String result =TenantCenter.findTenantsByUserId(userId);
SDKJsonResponse jr = SDKResultUtils.getData(result, "status", "msg", "tenants");
```

响应示例：

```
{
"status": "1",
"tenants":[{
…// 租户信息
},{
}]
}
```

4.1.6 根据租户 Id 查询指定租户信息

接口：com.yonyou.iuap.tenant.sdk.TenantCenter

接口类型：rest

方法名：getTenantById

方法说明：根据租户 Id 查询指定租户信息

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
tenantId	String	租户 Id	true	

响应参数：

参数名称	参数类型	参数说明	示例值
status	String	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
tenant	Tenant	租户	

请求示例：

```
String result =TenantCenter.getTenantById (tenantId);
SDKJsonResponse jr = SDKResultUtils.getData(result, "status", "msg", "tenant");
```

响应示例：

```
{
"status" : 1,
"tenant" :{
//  租户信息
}
}
```

4.1.7 根据企业 ID 获取企业信息

接口地址：com.yonyou.yht.sdk.EnterpriseCenter

接口类型：rest

请求方法：getEnInfo

方法说明：根据企业 Id 查询指定企业信息

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值

enid	String	企业 ID	true	

响应参数：

参数名称	参数类型	参数说明	示例值
status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
enterprise	EnterpriseInfo	企业信息列表；status 为 0 时无该字段	
msg	String	失败信息的描述；当 status 为 1 时，无该字段	

请求示例：

```
String result =EnterpriseCenter.getEnInfo(enid);
SDKJsonResponse jr = SDKResultUtils.getData(result, "status", "msg", "enterprise");
```

响应示例：

```
{
  "status" : 1,
  "enterprise" :{
   //  企业信息
}
}
```

4.1.8 根据用户 ID 获取该用户所管理的企业信息

接口地址：com.yonyou.yht.sdk.EnterpriseCenter

接口类型：rest

请求方法：getMemberEnListByUid

方法说明：根据用户 ID 获取该用户所管理的企业信息

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值

userId	String	用户 ID	true	

响应参数：

参数名称	参数类型	参数说明	示例值
status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
enterprises	List<EnterpriseInfo>	企业信息列表；status 为 0 时无该字段	
msg	String	失败信息的描述；当 status 为 1 时，无该字段	

请求示例：

```
String result =EnterpriseCenter.getMemberEnListByUid(userId);
SDKJsonResponse jr = SDKResultUtils.getData(result, "status", "msg", "enterprises");
```

响应示例：

```
{
"status" : 1,
"enterprises" :[
{
  //  企业信息
}}
}
```

4.1.9 根据账号密码获取 accesstoken

接口地址：com.yonyou.yht.sdk.UserCenter

接口类型：rest

请求方法：generateAccessToken

方法说明：根据账号密码获取 accesstoken

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
userName	String	用户 ID	true	
md5Password	String	MD5 密码	true	
shaPassword	String	Sha 密码	true	
multiLogin	boolean	是否支持多设备登录	true	

响应参数：

参数名称	参数类型	参数说明	示例值
status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
result	String	Accesstoken 信息	

请求示例：

```
String result = UserCenter.generateAccessToken(userName,
                                    md5Password, shaPassword, multiLogin);
SDKJsonResponse jr = SDKResultUtils.getData(result, "status", "result");
```

响应示例：

```
{
  "status" : 1,
  "result" :{
"accessToken":"f2a2314a-1fdd-41e3-9119-4b2e8bb63ca5",
"refreshToken":"f2a2314a-1fdd-41e3-9119-4b2e8bb63ca5",
"expires_in":"86400"
}
  }
```

4.1.10 刷新登录账号的 accesstoken

接口地址：com.yonyou.yht.sdk.UserCenter

接口类型：rest

请求方法：refreshAccessToken

方法说明：刷新登录账号的 accesstoken

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
refresh_token	String	刷新 token	true	

响应参数：

参数名称	参数类型	参数说明	示例值
status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
result	String	刷新后的 Accesstoken 信息	

请求示例：

```
String result = UserCenter.refreshAccessToken (refresh_token);
SDKJsonResponse jr = SDKResultUtils.getData(result, "status", "result");
```

响应示例：

```
{
  "status" : 1,
  "result" :{
"accessToken":"f2a2314a-1fdd-41e3-9119-4b2e8bb63ca5",
"refreshToken":"f2a2314a-1fdd-41e3-9119-4b2e8bb63ca5",
"expires_in":"86400"
}
  }
```

4.1.11 销毁 accesstoken

接口地址：com.yonyou.yht.sdk.UserCenter

接口类型：rest

请求方法：destroyAccessToken

方法说明：销毁客户端登录获取到的 accesstoken

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
accesstoken	String	accesstoken	true	

响应参数：

参数名称	参数类型	参数说明	示例值
status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
result	String	accessToken 销毁提示	

请求示例：

```
String result =UserCenter.refreshAccessToken(refresh_token);
SDKJsonResponse jr = SDKResultUtils.getData(result, "status", "result");
```

响应示例：

```
{
  "status" : 1,
  "result" : ""

}
```

4.1.12 根据 accesstoken 获取临时 token

接口地址：com.yonyou.yht.sdk.UserCenter

接口类型：rest

请求方法：genTokenByAccessToken

方法说明：根据 accesstoken 获取临时 token

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值

accessToken	String	accessToken	true	

响应参数：

参数名称	参数类型	参数说明	示例值
status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
result	String	token 的内容	

请求示例：

```
String result =UserCenter.genTokenByAccessToken(accessToken);
SDKJsonResponse jr = SDKResultUtils.getData(result, "status", "result");
```

响应示例：

```
{
"status" : 1,
"result" :{
"token":"f2a2314a-1fdd-41e3-9119-4b2e8bb63ca5"
}
}
```


4.1.13 给用户发送验证码

接口地址：com.yonyou.yht.sdk.UserCenter

接口类型：rest

请求方法：sendValidateCode

方法说明：给指定用户发送验证码，可指定给手机发送或者是给邮箱发送。

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
userId	String	用户 Id	true	

type	String	手机或邮箱	true	mobile/email

响应参数：

参数名称	参数类型	参数说明	示例值
status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
msg	String	发送结果	

请求示例：

```
String result =UserCenter.sendValidateCode (userid,type);
SDKJsonResponse jr = SDKResultUtils.getData(result, "status", "msg");
```

响应示例：

```
{
  "status" : 1,
  "msg" : "短信发送成功"
}
```

4.1.14 校验验证码

接口地址：com.yonyou.yht.sdk.UserCenter

接口类型：rest

请求方法：validateCode

方法说明：用户收到验证码后在页面输入验证码，校验用户输入的验证码是否正确。

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
type	String	手机或邮箱	true	mobile/email
userId	String	用户 Id	true	
code	String	验证码	true	

contact	String	手机号或邮箱，userId为空时生效	false	

响应参数：

参数名称	参数类型	参数说明	示例值
status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
sid	String	后续做安全有关的 sessionid，只能使用一次，只能针对相应的 user 进行安全接口调用，无 userId	

请求示例：

```
String result =UserCenter.validateCode(type, userId, code, contact);
SDKJsonResponse jr = SDKResultUtils.getData(result, "status", "sid");
```

响应示例：

```
{
  "status" : 1,
  "sid" : "5aa925958acd671d0323f97ff740a832"
}
```

4.1.15 修改密码

接口地址：com.yonyou.yht.sdk.UserCenter

接口类型：rest

请求方法：modifyPassword

方法说明：输入旧密码确认后将用户密码替换为新输入的密码。

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
userId	String	用户 Id	true	
shaPassword	String	旧密码的 sha-1 值	true	

md5Password	String	旧密码的 md5 值	true	
shaNewPassword	String	新密码的 sha-1 值	false	
md5NewPassword	String	新密码的 md5 值		
sid	String	验证码校验接口返回的 sid		

响应参数：

参数名称	参数类型	参数说明	示例值
status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
msg	String	更新结果	

请求示例：

```
String result =UserCenter.modifyPassword(userId, shaPassword,
                                md5Password, shaNewPassword, md5NewPassword, sid);
SDKJsonResponse jr = SDKResultUtils.getData(result, "status", "sid");
```

响应示例：

```
{
  "status" : 1,
  " msg " : "修改成功"
}
```

4.1.16 重置密码

接口地址：com.yonyou.yht.sdk.UserCenter

接口类型：rest

请求方法：resetPassword

方法说明：输入收到的验证码和新密码，直接将密码置位新设置的密码

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
userId	String	用户 Id	true	
shaNewPassword	String	新密码的 sha-1 值	false	
md5NewPassword	String	新密码的 md5 值		
sid	String	验证码校验接口返回的 sid		

响应参数：

参数名称	参数类型	参数说明	示例值
status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
msg	String	更新结果	

请求示例：

```
String result =UserCenter.resetPassword(userId, shaNewPassword, md5NewPassword, sid);
SDKJsonResponse jr = SDKResultUtils.getData(result, "status", "sid");
```

响应示例：

```
{
  "status" : 1,
  " msg " : "重置成功"
}
```

4.1.17 修改手机号或邮箱

接口类：com.yonyou.yht.sdk.UserCenter

接口类型：rest

请求方法：modifyContact

方法说明：修改手机号或邮箱

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
userId	String	用户 id	true	
contact	String	手机号或邮箱	true	
sid	String	验证码校验接口返回的 sid（参加校验验证码）	true	

响应参数：

参数名称	参数类型	参数说明	示例值
status	String	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
msg	String	更新结果或异常提示	

请求示例：

```
String result =UserCenter.modifyContact (String userId, String contact, String sid)
```

响应示例：

```
{
  "status" : 1,
  " msg " : "修改成功"
}
```

4.1.18 快速批量增加用户

接口类：com.yonyou.yht.sdk.UserCenter

接口类型：rest

请求方法：quickAddUsers

方法说明：批量增加用户（最多 1000 个）

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
sysId	String	系统编码，表示是属于那个产品的用户（例如：ipu，hr_cloud）	true	
jsonStr	UserVO	用户信息，必有的值 userMobile 或者 userEmail 至少一个不能为空；可选 userPassword，密码可不传，不传的话会被设置为默认密码；需要传 systId，产品标识。	true	
showUser	boolean	在返回数据中，是否显示用户信息。true 显示，false 不显示，默认是不显示		

响应参数

参数名称	参数类型	参数说明	示例值
status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
msg	String	更新结果	
mobile	String	返回传入的 userMobile	
email	String	返回传入的 userEmail	
userId	String	用户 id，当用户匹配成功或者新建的用户有此参数	

flag	int	当用户匹配成功或者新建的用户有此参数,0 ：匹配到已存在的用户，1：新建的用户	
user	String	user 用户信息，当传入的参数 showUser 为 true 有此参数	

响应示例：

备注：datas 中的数据顺序跟传入的 users 的列表顺序是一样的


```
{
  "status" : 1,
  "datas" : [ {
    "status" : 0,
    "msg" : "手机号格式不正确",
    "email" : "test998879@mytestzp.com",
    "mobile" : "1523333"
  },
 {
    "status" : 1,
    "userId" : "0c79ff01-dcce-4531-8048-2333a6d35a5b",
    "user" : {
      "userId" : "0c79ff01-dcce-4531-8048-2333a6d35a5b",
      "userCode" : "YHT-1023-9811546318128423",
      "userName" : "test23",
      "userAvator" : null,
      "userAvatorNew" : null,
      "userBigAvatorNew" : null,
      "userSmallAvatorNew" : null,
      "sex" : -1,
      "birthday" : null,
      "address" : null,
      "industry" : null,
      "position" : null,
      "userMobile" : null,
```

```
      "userEmail" : "zptest998883@mytestzp.com",
      "sysId" : "yzt4",
      "sysName" : null,
      "sysEnName" : null,
      "addr" : null,
      "timeZone" : null,
      "lang" : null,
      "postCode" : null,
      "fromYht" : false,
      "sysMergeIcon" : null,
      "currentUser" : false,
      "disableModify" : null,
      "userNameCensor" : false,
      "countryCode" : null,
      "privileged" : false,
      "mergeUser" : false
    },

    "msg" : "zptest998883@mytestzp.com 联系方式已经存在",
    "flag" : 0
    "email" : "zptest998883@mytestzp.com",
    "mobile" : null
  }
]
}
```


4.1.19 登录校验接口

接口类：com.yonyou.yht.sdk.UserCenter

接口类型：rest

请求方法：verifyUser

方法说明：登录校验接口

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
userName	String	登录名（必填）	true	
userPassword	String	明文密码（SDK 内部会进行加密）	true	

响应参数：

参数名称	参数类型	参数说明	示例值
status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
msg	String	更新结果	

请求示例：

```
String result = UserCenter.verifyUser("13811394465", "q1111111", "yht");
```

响应示例：


```
{
"msg": "",
"status": 1,
"user": {
"userAvatorNew": "",
"loginType": "YHT",
"sysId": "hr_cloud",
"userBigAvatorNew": "",
"auditAction": "",
"source": "",
"userCode": "HRCxxx8264",
"sid": 0,
"registerSource": "",
"validateMobile": false,
"userMobileCountrycode": "86",
```

```
"userMobile": "137xxxx8264",
"userActivate": true,
"userSmallAvatorNew": "",
"userEmail": "",
"lang": "",
"registerDate": "2018-04-09 02:01:48",
"outDate": "",
"needMerge": false,
"verified": false,
"timeZone": "",
"userName": "XXX",
"userId": "a83f68fc-xxx0aa747",
"validateEmail": false,
"realName": "",
"userAvator": "",
"censor": false,
"mailValidatect": 0,
"account": "",
"status": 0,
"ts": {
"date": 9,
"hours": 2,
"seconds": 29,
"month": 3,
"nanos": 0,
"timezoneOffset": -480,
"year": 118,
"minutes": 2,
"time": 1523210549000,
"day": 1
}
}
}
```

4.1.20 分页获取租户下用户列表

接口：com.yonyou.iuap.tenant.sdk.UserCenter

接口类型：rest

方法名：getUserList

方法说明：获取租户下的用户分页数据。

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
tenantId	String	租户 ID	true	
ps	String	一页显示多少数据，默认为 20		
pn	String	第几页，默认为 1		
sortType	String	默认值 auto		

响应参数：

参数名称	参数类型	参数说明	示例值
status	String	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
users	List<Tenant>	租用户列表	

请求示例：

```
String result =TenantCenter.getUserList(userId,1,20,null);
```

响应示例：

```
{
  "status" : 1,
  "users" : {
    "first" : true,
    "firstPage" : true,
    "last" : false,
```

```
    "lastPage" : false,
    "number" : 0,
    "numberOfElements" : 0,
    "size" : 10,
    "totalElements" : 1001,
    "totalPages" : 101,
    "content" : [ {
      "userId" : "07d4b98a-950b-429c-af77-628164291743",
      "userName" : "测试用户 6007",
      "userCode" : "yhttest6007",
      "userAvator" : "",
      "userMobile" : "13010026007",
      "userEmail" : "yhttestt6007@yonyou.com",
      "needMerge" : false,
      "verified" : false,
      "validateEmail" : false,
      "validateMobile" : false,
      "sysId" : null,
      "registerDate" : "2017-03-21 13:56:30",
      "loginTs" : 1492581913125
    }…]
  }
}
```


4.1.21 根据租户 id 获取租户下所有管理员数据

接口：com.yonyou.iuap.tenant.sdk.UserCenter

接口类型：rest

方法名：getTenantAdminByTenantId

方法说明：获取租户下的用户分页数据。

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值

tenantId	String	租户 ID	true	

响应参数：

参数名称	参数类型	参数说明	示例值
status	String	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
tenantUsers	List<TenantUser>	用户列表	

请求示例：

```
String result =UserCenter.getTenantAdminByTenantId("dls2l3kn");
```

响应示例：

```
{
        "tenantUsers": [
                {
                        "userId": "61bee285-11d1-4a5f-a35d-405ddad36acc",
                        "userName": "eiap",
                        "userCode": "eiap",
                        "userAvator": "",
                        "userMobile": "15727396814",
                        "userEmail": "zhangbing1@yonyou.com",
                        "needMerge": false,
                        "verified": false,
                        "validateEmail": false,
                        "validateMobile": false,
                        "userPassword": null,
                        "sysId": null,
                        "registerDate": "2017-02-27 00:50:35",
                        "loginTs": 1496904522678
                },
                {
                        "userId": "e7643d78-bc19-4c4a-bbc4-94762251f257",
                        "userName": "eiapmanager7",
                        "userCode": "eiapmanager7",
                        "userAvator": "",
```

```
                "userMobile": "",
                "userEmail": "eiapmanager7@yonyou.com",
                "needMerge": false,
                "verified": false,
                "validateEmail": false,
                "validateMobile": false,
                "userPassword": null,
                "sysId": null,
                "registerDate": "2017-04-26 14:31:50",
                "loginTs": 1496904522681
            }
        ],
        "status": 1
}
```


4.1.22 将用户加入指定租户

接口：com.yonyou.iuap.tenant.sdk.UserCenter

接口类型：rest

方法名：addToTenant

方法说明：根据用户 id 查询该用户在该产品可登录的租户。

如果租户管理员开启授权，该用户没有授权该产品，那么该租户不可登录。

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
tenantId	String	租户 ID	true	
userType	int	用户类型，1 为租户管理员，其它为普通用户	true	
Pks	String[]	用户 id 数组	true	

响应参数：

参数名称	参数类型	参数说明	示例值

Status	String	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
Msg	String	添加关联成功	

请求示例：

```
String result =UserCenter.addToTenant("vjjry6i0", 3, ids.toArray(new String[ids.size()]));
```

响应示例：

成功返回：

```
{
  "errorIds" : [ ],
  "status" : 1,
  "msg" : "添加关联成功"
}
```

失败返回

```
 {
     "status": 0,
     "msg": "添加失败"
 }
```

部分成功

```
{
    "status":1
    "errorIds":["userid1","userid2"]/*为没有成功的 userid*/
}
```

4.1.23 解除租户和用户之间的关联

接口类：com.yonyou.iuap.tenant.sdk.UserCenter

接口类型：rest

请求方法：removeFromTenant

方法说明：解除租户和用户之间的关联

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
tenantId	String	租户 Id	true	
pks	String[]	用户的 userID 集合，最多 100	true	

响应参数：

参数名称	参数类型	参数说明	示例值
status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
msg	String	添加成功或添加失败	

请求示例：

```
String result =UserCenter.removeFromTenant(tenantId, String[] pks);
SDKJsonResponsejr = SDKResultUtils.getData(result, "status", "msg");
```

响应示例：

```
{
   "status" : 1,
   "msg" : "解除关联成功"
}
```

4.1.24 获取产品下可登录租户列表

接口地址：com.yonyou.iuap.tenant.sdk.TenantCenter

接口类型：rest

请求方法：getCanLoginTenants

方法说明：获取产品下可登录租户

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
userId	String	用户 Id	true	

resCode	String	产品编码	true	
productLineCode	String	产品线编码	false	diwork和u8c3.0;不需要时传null

响应参数：

参数名称	参数类型	参数说明	示例值
status	int	请求的状态，如果为0，表示失败，如果为1，表示成功	
tenants	List<Tenant>	租户列表	

请求示例：

```
String result = TenantCenter.getCanLoginTenants(userId, resCode, productLineCode);
SDKJsonResponse jr = SDKResultUtils.getData(result, "status", "tenants");
```

响应示例：

```
{
  "status" : 1,
  "tenants" : [ {
    // 租户信息
  } ]
}
```

4.1.25 根据账号判断是否是管理员

接口地址：com.yonyou.iuap.tenant.sdk.UserCenter

接口类型：rest

请求方法：isAdminNew

方法说明：根据账号判断是否是管理员

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
tenantId	String	租户 Id	true	
userId	String	用户 Id	true	

响应参数：

参数名称	参数类型	参数说明	示例值
status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
flag	int	表示是否是管理员，1 为是，0 为否	

请求示例：

```
String result =UserCenter.isAdminNew(tenantId,userId);
SDKJsonResponse jr = SDKResultUtils.getData(result, "status", "flag");
```

响应示例：

```
{
  "status" : 1,
  "flag" : 0
}
```

4.1.26 搜索租户下获取用户列表(分页)

接口地址：com.yonyou.iuap.tenant.sdk.UserCenter

接口类型：rest

请求方法：searchTenantUserList

方法说明：分页搜索该租户下、产品的用户

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
tenantId	String	租户 Id	true	

resCode	String	产品编码	true	
ps	String	分页大小	false	
pn	String	第几页		
searchcode	String	关键字，支持（用户名称、手机号、邮箱）		
sortType	String	排序，默认为 null		

响应参数：

参数名称	参数类型	参数说明	示例值
status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
users	Page<User>	用户列表信息	

请求示例：

```
String result =UserCenter.searchTenantUserList(tenantId, resCode, ps, pn,
                                    searchcode, sortType);
SDKJsonResponse jr = SDKResultUtils.getData(result, "status", "users");
```

响应示例：

```
{
  "status" : 1,
  "users" : {
   // Page 包装的用户信息
  }
}
```

4.1.27 批量添加用户，如果存在的用户直接添加关联关系

接口地址：com.yonyou.iuap.tenant.sdk.UserCenter

接口类型：rest

请求方法：addUsersAndRelations

方法说明：批量添加用户，如果存在的用户直接添加关联关系，最多 50 个

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
tenantId	String	租户 Id	true	
systemCode	String	系统编码（例如：ipu、hr_cloud）	true	
source	String	来源（例如 :UserSource.LOCAL 、UserSource.NC_PRIVATE_CLOUD）	false	
cuser	String	操作用户的 ID		
jsonStr	String	批量添加的 json 串		

jsonStr 示例：

```
{
  "users": [
    {
      "userCode": "test1",
      "userName": "测试用户 1",
      "userMobile": "13412343431",
      "userEmail": "test1@yonyou.com",
      "para1": "random"//添加该参数，那么新创建的用户为随机密码
    },
    {
      "userCode": "test2",
      "userName": "测试用户 2",
      "userMobile": "13412343432",
      "userEmail": "test2@yonyou.com"
    }
  ]
}
```

响应参数：

参数名称	参数类型	参数说明	示例值
status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
msg	String	添加成功或添加失败	
users	List<User>	用户列表信息	

请求示例：

```
String result =UserCenter.addUsersAndRelations(tenantId, systemCode, source,cuser,jsonStr);
SDKJsonResponse jr = SDKResultUtils.getData(result, "status", "msg","users");
```

响应示例：

```
{
  "status" : 1,
  "msg":"添加成功",
  "users" : [
  {// 用户信息}
  ]
}
```


4.1.28 解除租户和用户之间的关联

接口地址：com.yonyou.iuap.tenant.sdk.UserCenter

接口类型：rest

请求方法：removeFromTenant

方法说明：解除租户和用户之间的关联

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
tenantId	String	租户 Id	true	

pks	String[]	用户的 userID 集合，最多 100	true	

响应参数：

参数名称	参数类型	参数说明	示例值
status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
msg	String	添加成功或添加失败	

请求示例：

```
String result =UserCenter.removeFromTenant(tenantId, String[] pks);
SDKJsonResponse jr = SDKResultUtils.getData(result, "status", "msg");
```

响应示例：

```
{
  "status" : 1,
  "msg":"添加成功"
}
```

4.1.29 产品开通

接口类：com.yonyou.iuap.tenant.sdk.TenantCenter

接口类型：rest

请求方法：StringBathOpenApp

方法说明：产品开通

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
userId	String	用户 Id	true	
resCode	String	产品编码	true	
productLineCode	String	产品线编码	false	diwork 和 u8c3.0;不需要时传 null

响应参数：

参数名称	参数类型	参数说明	示例值
status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
success	JSONArray	成功列表	
errors	JSONArray	错误列表	
errMsg	JSONArray	错误信息	

请求示例：

```
TenantCenter.StringBathOpenApp(
    "[{\"beginDate\":\"2015-12-10 12:00:00\",\"resCode\":\"retail\",\"tenantId\":\"xbgevu1a\",\"endDate\":\"2018-11-11 01:10:00\"},"
    +"{\"beginDate\":\"2016-05-0112:00:00\",\"resCode\":\"u8c\",\"tenantId\":\"xbgevu1a\",\"endDate\":\"2018-121-01 01:10:00\"}]" );
```

响应示例：

```
{
  "success" : [ {
    "tenantId" : "xbgevu1a",
    "resCode" : "retail"
  }, {
    "tenantId" : "xbgevu1a",
    "resCode" : "u8c"
  }],
  "errors" : [ ],
  "errMsg" : [ ],
  "sucessMsg" : [ "xbgevu1a 开通 retail 成功", "xbgevu1a 开通 u8c 成功"]
}
```

4.1.30 查询产品剩余使用时间

接口类：com.yonyou.iuap.tenant.sdk.TenantCenter

接口类型：rest

请求方法：getRemainTime

方法说明：查询产品剩余使用时间

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
sysid	String	产品 code	true	
tenantId	String	租户 id	true	

响应参数：

参数名称	参数类型	参数说明	示例值
status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
remainTime	Long	剩余时间	

请求示例：

```
TenantCenter.getRemainTime (String sysid, String tenantId);
```

响应示例：

```
{
  "status" : 1,
  "remainTime" : 1626278346000
}
```

4.1.31 新增租户

接口类：com.yonyou.iuap.tenant.sdk.TenantCenter

接口类型：rest

请求方法：addTenant

方法说明：新增租户

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
tenantCode	String	租户 code	true	
tenantName	String	租户名称	true	
tenantAddress	String	租户地址	true	
tenantTel	String	联系方式		
tenantEmail	String	邮箱		
tenantFullname	String	租户全名		
orgCode	String	机构编码		
source	String	创建来源		
team	String	如果创建的是团队需要传 "team":"1"		

响应参数：

参数名称	参数类型	参数说明	示例值
status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
tenant	Tenant	租户信息	

请求示例：

```
TenantCenter.addTenant(Map<String, String> params);
```

响应示例：

```
    {
   "status" : 1,
"tenant" : {
   "tenantId" : "ga79pctf",
        ...
}
```

4.1.32 激活产品 API-openApp

注：此接口需要产品**提供**，并在上架时提供给友户通审核人员。

接口地址：openApp

接口类型：rest/POST

方法说明：校验激活码，校验签名，开通服务（续费，架构），返回服务开通结果

请求参数：

参数名称	参数类型	参数说明
action	String	开通动作，目前仅为 active，后面会添加其他行为，如续费 renewInstance， 商品过期 expiredInstance， 商品释放 releaseInstance
activationCode	String	激活码
tenantId	String	企业帐号 id
enterpriseId	String	企业 id（开通类型为企业时会传递该参数）
userId	String	使用激活码的用户 id
openType	int	开通类型：仅企业账号开通 0，仅企业开通 1，企业帐号支持用户开通 10，企业支持用户开通 11
resCode	String	商品资源码，服务商提供的 openApp 接口回写产品开通状态的时候需要回传
orderId	String	订单号

orderCid	String	子订单 Id
orderBizId	String	预留字段
skuId	String	产品版本，服务商提供的 openApp 接口需要根据 skuId 匹配具体的产品
accountQuantity	int	购买数量、人数，由产品属性确定，有的产品可以选择人数和数量
expiredOn	String	过期时间(yyyy-MM-ddHH:mm:ss)
productId	String	产品 Id（云商务上的产品开通时会传递该参数）
productName	String	产品名称（云商务上的产品开通时会传递该参数）
companyName	String	企业名称（开通类型为企业时会传递该参数）
companyShortName	String	企业简称（开通类型为企业时会传递该参数）
lease	String	订单租期（单位月，使用 expireOn 取代）
email	String	客户邮箱
mobile	String	客户手机
ts	String	时间戳，加签验签使用
instanceId	String	续费，商品过期，商品释放时会传递该参数

响应参数：

参数名称	参数类型	参数说明
status	int	0：失败（参数错误，或其他原因，具体放到 msg 中），1：成功，2：立即开通，3：产品推送开通状态 说明：如果开通出错返回 0 ；如果请求成功，不是立即开通，返回 3（后续调用 openResCallBack 接口）；如果请求成功且已开通，返回 2 ；

msg	String	错误信息(长度限制 1000 以内)
data	String	成功时带有的数据：字符串（可以 json 格式）

接口示例：

```
@RestController
public class AppController {
    @PostMapping("openApp")
    public String openApp(HttpServletRequest req){
    }
}
```

响应示例：

```
{
"status" : 1,
"msg" :
}
```

4.1.33 产品开通状态回写

注：如果 openApp 不是立即返回开通结果时使用

接口地址：/app/j/openResCallBack

接口类型：rest

方法说明：如果产品开通很快，那么可以直接返回结果 status=2，而不用再推送状态，否则返回 status=3,开通后需要调用接口同步 openResCallBack。

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
activeCode	String	激活码	true	
resCode	String	产品简称	true	
tenantId	String	企业帐号 id	true	

备注：请求需要加签

响应参数：

参数名称	参数类型	参数说明	示例值
status	int	检查结果,如果为 0，表示失败，如果为 1，表示成功	
msg	String	错误信息	

请求示例：

```
String result =AppCenter.openResCallBack(activeCode,tenantId, resCode);
```

响应示例：

```
{
  "status" : 1,
  "msg":"开通成功"
}
```

4.1.34 增加身份

接口：com.yonyou.iuap.tenant.sdk.UserCenter

接口类型：rest

接口说明：分页获取用户列表

方法名：addUserIdentityToTenant

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
tenantId	String	租户 id	true	
resCode	String	产品编码	true	diwork
userType	int	用户在租户下类型填 3	false	3
pks	String[]	用户 id 集合，最多 100	true	
authorizerId	String	授权者 id	false	
accessToken	String	Token 不能空	true	

响应参数：

参数名称	参数类型	参数说明	示例值
status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
msg	String	返回信息：添加关联成功、添加失败	

请求示例：

```
String result = com.yonyou.yht.sdk.UserCenter.addUserIdentityToTenant(tenantId,resCode,userType,pks,authorizerId,accessToken);
```

响应示例：

```
{
    "status": 1,
    "msg": "添加关联成功"
}
```

4.1.35 批量停用用户身份

接口：com.yonyou.iuap.tenant.sdk.UserCenter

接口类型：rest

接口说明：批量停用用户身份

方法名：batchStopUserIdentity

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
tenantId	String	租户 id	true	
resCode	String	产品编码	true	diwork
userIds	String[]	友户通用户 id 数组	true	

响应参数：

参数名称	参数类型	参数说明	示例值
status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
msg	String	返回信息：	更新成功

请求示例：

```
String result = com.yonyou.yht.sdk.UserCenter.batchStopUserIdentity (tenantId,resCode,userIds);
```

响应示例：

```
{
    "status": 1,
    "msg": "更新成功"
}
```

4.1.36 批量启用用户身份

接口：com.yonyou.iuap.tenant.sdk.UserCenter

接口类型：rest

接口说明：批量启用用户身份

方法名：batchEnableUserIdentity

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
tenantId	String	租户 id	true	
resCode	String	产品编码	true	diwork
userIds	String	友户通用户 id 数组	true	

响应参数：

参数名称	参数类型	参数说明	示例值

status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
msg	String	返回信息：	更新成功

请求示例：

```
String result = com.yonyou.yht.sdk.UserCenter.batchEnableUserIdentity (tenantId,resCode,userIds);
```

响应示例：

```
{
    "status": 1,
    "msg": "更新成功"
}
```

4.1.37 删除身份

接口：com.yonyou.iuap.tenant.sdk.UserCenter

接口类型：rest

接口说明：删除身份

方法名：removeIdentityFromTenant

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
tenantId	String	租户 id	true	
identityType	String	身份类型值默认 1，普通员工	true	1
resCode	String	产品编码	true	diwork
userIds	List<String>	友户通用户 id 列表	true	

响应参数：

参数名称	参数类型	参数说明	示例值

status	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
msg	String	返回信息	操作成功

请求示例：

```
String result = com.yonyou.yht.sdk.UserCenter.removeIdentityFromTenant(tenantId,identityType,resCode,userIds);
```

响应示例：

```
{
    "status": 1,
    "msg": "操作成功"
}
```

4.1.38 关键字搜索用户身份

接口：com.yonyou.iuap.tenant.sdk.UserCenter

接口类型：rest

接口说明：分页查询用户身份

方法名：searchUserIdentity

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
tenantId	String	租户 id	true	
resCode	String	产品编码	true	diwork
keyword	String	搜索关键字可空	false	
extFilters	Map	预留过滤字段空	false	
userTypes	List<Ingteger>	身份类型列表，可空	false	
pn	int	页码默认 1	true	1

ps	int	每页大小默认 10	true	10

响应参数：

参数名称	参数类型	参数说明	示 例值
status	int	请求的状态，如果为 0，表示失败，如果为，表示成功	
data	JsonObject		见示例

请求示例：

```
String result = com.yonyou.yht.sdk.UserCenter.searchUserIdentity (tenantId, resCode, keyword, extFilters, userTypes, pn, ps);
```

响应示例：

```
{
  "status" : 1,
  "data" : {
    "pageSize" : 10,
    "pageNum" : 1,
    "total" : 1,
    "totalPage" : 1,
    "content" : [ {
      "tenant_id" : "kf2cd32u",
      "create_user_id" : "0a056967-60c9-4b71-ae93-ac7d5ff16cd8",
      "user_email" : "cewerw@test1988.com",
      "user_mobile_country_code" : "86",
      "user_type" : 1,
      "user_code" : "YHT-341-5241530698718326",
      "user_id" : "0a056967-60c9-4b71-ae93-ac7d5ff16cd8",
      "user_mobile" : "17865198680",
      "user_name" : "xccc",
      "id" : 1166062689341696,
      "res_code" : "diwork",
```

```
        "status" : 1
      } ],
      "domainTotal" : 0
    }
}
```

4.1.39 分页查询用户身份

接口：com.yonyou.iuap.tenant.sdk.UserCenter

接口类型：rest

接口说明：分页获取用户列表

方法名：getUserIdentityByTenantId

请求参数：

参数名称	参数类型	参数说明	是否必填	示例值
tenantId	String	租户 id	true	
resCode	String	产品编码	true	diwork
ps	String	每页大小默认 20	false	
pn	String	页码默认 1		
searchCode	String	身份类型列表，可空	false	
sortType	String	页码默认 1	false	1
userType	String	每页大小默认 20	false	20
status	String	用户身份状态	false	

响应参数：

参数名称	参数类型	参数说明	示例值

state	int	请求的状态，如果为 0，表示失败，如果为 1，表示成功	
data	JsonObject		

String result = com.yonyou.yht.sdk.UserCenter.getUserIdentityByTenantId (tenantId, resCode, ps,pn, searchCode, sortType, userType, status);

响应示例：


```
{
  "state": 1,
  "data": {
    "pageSize": 1,
    "pageNum": 0,
    "total": 101,
    "totalPage": 101,
    "content": [
      {
        "id": 1250569704771840,
        "yhtUserId": "65d7b902-9fb0-421e-9707-65a137648a91",
        "tenantId": "qmc9xo0p",
        "tenantName": null,
        "typePro": 0,
        "yxyTenantId": 1247781491577088,
        "resCode": "diwork",
        "userType": 1,
        "customerId": null,
        "status": 0,
        "uts": 1571719866000,
        "deleted": 0,
        "createUserId": "8b99f589-bc47-4c3a-bf1c-13d78caa20b0",
        "userName": "史佳琦",
        "userMobile": "18697367037",
        "userCode": null,
        "userEmail": "shijqi@yonyou.com",
```

```
            "json": null,
            "docIds": null,
            "yxyUserId": "935ed24956c24272be06ac23c3376ce8"
          }
        ],
        "domainTotal": 0
      }
    }
```


4.2 接入友户通开发场景

4.2.1 单点登录接入

接入单点登录需要配置 web.xml 拦截器和在 sdk.properties 文件中添加配置。

1、添加监听和拦截

1) 配置 Web.xml

设置过滤器，Filter 部分要放在配置文件的所有 filter 之前。

```
<filter>
    <filter-name>CAS Single Sign Out Filter</filter-name>
    <filter-class>com.yonyou.yht.web.cas.sso.SingleSignOutFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>CAS Single Sign Out Filter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
<filter>
    <filter-name>CAS Authentication Filter</filter-name>
    <filter-class>com.yonyou.yht.web.cas.AuthenticationFilter2</filter-class>
</filter>
<filter>
    <filter-name>CAS Validation Filter</filter-name>
    <filter-class>com.yonyou.yht.web.cas.ProxyReceivingTicketValidationFilter</filter-class>

    <init-param>
        <param-name>encoding</param-name>
        <param-value>UTF-8</param-value>
    </init-param>
</filter>
<filter>
    <filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>
    <filter-class>org.jasig.cas.client.util.HttpServletRequestWrapperFilter</filter-class>
</filter>
<filter>
    <filter-name>CAS Assertion Thread Local Filter</filter-name>
    <filter-class>org.jasig.cas.client.util.AssertionThreadLocalFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>CAS Authentication Filter</filter-name>
```

```
    <url-pattern>/*</url-pattern>
</filter-mapping>
<filter-mapping>
    <filter-name>CAS Validation Filter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
<filter-mapping>
    <filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
<filter-mapping>
    <filter-name>CAS Assertion Thread Local Filter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

2） 配置 sdk.properties

如果需要单点登录的过滤器放过某些地址的请求的话，可在 sdk.properties 中配置如下参数：

```
   ##不需要拦截的模式名称支持 CONTAINS（包含）、MUTICONTAINS（多个，逗号分隔的包含）、REGEX（正则表达式）、EXACT（相等）、MUTIREGEX（多个正则表达式，逗号分割）;
ignoreUrlPatternType=MUTIREGEX
##不需要拦截的模式表达式
ignorePattern=.*\\.js,.*\\.css,.*\\.jsx,/rest/.*,/style/.*,/index\\.html,/controller/register/.*
```

如果需要为单点登录的过滤器配置定制的过滤条件类的话，可在 sdk.properties 中配置如下参数：标红的类需要自己去实现接口 IRequestMatcherStrategy（仅需实现方法 matches，返回 true 代表放过改请求，不进行登录拦截），**推荐使用自定义方法进行拦截判断**。如下示例

```
   ignoreUrlClassPath=com.yonyou.yht.web.cas.MutiContainsPatternUrlPatternMatcherStrategy
```

3） 配置会话存储

会话存储支持 redis、JVM 内存 MAP 两种方式，默认使用 JVM 内存 MAP 来存储，建议使用 Redis 方式，如果使用 Redis 方式请在 sdk.properties 里配置：

sessionStoreClass=com.yonyou.yht.cache.RedisSessionStore

3、验证登录流程

1） 构建业务系统

构建简单的业务系统 HelloApp2

2） 浏览器访问测试系统

当前测试系统的 URL 地址为：http://localhost:8080/HelloApp2/test。

3） 浏览器重定向至友户通登录页面
重定向 URL 地址为：

http://idtest.yyuap.com/cas/login?sysid=test&service=http://localhost:8080/HelloApp2/test

4） 登录成功，返回测试系统
5） 登录失败，登录页面提示错误信息，如密码错误等。

4、获取登录后的用户信息

1） 获取登录用户信息
用户信息内容包括：

user 用户名

userCode 用户名称

userId 用户 id

authToken 认证 Token

```
    AttributePrincipal principal =
                          (AttributePrincipal)request.getUserPrincipal() ;
principal.getName();
principal.getAttributes();
```

2） 获取当前登录用户的 userId

```
    String userId = CasClientUtils.getLoginedUserId()
```

4.2.2 单点注销

根据操作类型注销可分为："前端注销"和"后端注销"两种类型；根据注销时产品后台是否需要处理自己的 Session 分为："管理自己 Session"和"无需操作"。

1、前端注销

首先需要按照单点登录章节配置好 Filter。

服务开发者需要在页面添加"注销"或"退出"按钮，用户在浏览器上点击按钮后需要向后端发送 logout 请求，URL 形式为 schema://servername/logout?SAMLRequest=true&service=servername，友户通的 Filter 会处理这个请求。

如果应用需要管理自己的 Session，参考下文管理自己的 Session。

2、后端注销

首先需要按照单点登录章节配置好 Filter，如果应用需要管理自己的 Session，参考下文管理自己的 Session。

3、管理自己 Session

产品有自己管理的 Session 时，需要处理两种情况。

A. 由前端用户自己触发的注销操作

前端“注销”或“退出”按钮，调用产品自己的后端接口进行注销，产品注销完自己的 session 后，在应用服务器端调用 CasClientUtils.destroyToken（java 程序），该方法将会返回一个 CAS 全局注销的 URL，产品需要重定向此 URL；浏览器接收到重定向请求后将会访问 URL，即实现了 CASServer 的注销。

B. 由 CASServer 触发的注销操作

此种情况是由于用户同时登录了两个产品，在其中一个产品里注销后，CASServer 会发送后端请求到另一个产品请求注销。这种情况下，应用的服务端需要做如下处理：

a) 实现钩子方法

需要继承两个类，分别为 ProxyReceivingTicketValidationFilter 和 SingleSignOutHandler，并重写或实现其中的钩子方法，友户通在登录和注销时，会调用实现的钩子方法。

i. 在登录时记录 ticket 和 session 的关系，需要继承 ProxyReceivingTicketValidationFilter，重写 saveLoginInfo 方法。

```
protected void saveLoginInfo(String ticket, Assertion assertion,
            HttpServletRequest request, HttpServletResponse response)
```

ii. 在退出时分别实现前端退出和后端退出，继承 SingleSignOutHandler，实现 frontLogout 和 backLogout 方法。

```
// 前端退出
protected void frontLogout(HttpServletRequest request,HttpServletResponse response)
// 后端退出
protected void backLogout(String ticket)
```

b) 修改 web.xml 中 SingleSignOutFilter 的参数

把自己实现的 SingleSignOutHandler 的子类作为 SingleSignOutFilter 的参数。

```
<filter>
```

```
<filter-name>CAS Single Sign Out Filter</filter-name>
<filter-class>com.yonyou.yht.web.cas.sso.SingleSignOutFilter</filter-class>
<init-param>
<param-name>自己实现的 SingleSignOutHandler 的子类</param-name>
<param-value>com.yonyou.yht.web.cas.sso.MySingleSignOutHandler</param-value>
</init-param>
</filter>
```

c） 修改 web.xml 中 ReceivingTicketValidationFilter

把 ReceivingTicketValidationFilter 替换为自己实现的 ReceivingTicketValidationFilter 的子类。

```
    <filter>
<filter-name>CAS Validation Filter</filter-name>
<filter-class>自己实现的 ReceivingTicketValidationFilter 的子类</filter-class>
<init-param>
<param-name>encoding</param-name>
<param-value>UTF-8</param-value>
</init-param>
</filter>
```

4.2.3 租户登录集成

租户集成方案是指登录后获取当前登录的租户信息和可以登录的租户列表信息，租户登录集成是在友户通单点登录集成方案的基础上实现的，请先集成友户通单点登录。

1、配置 Filter

Filter 部分也要放在配置文件的所有友户通的 Filter 之后，应用自己的 Filter 之前。

```
    <!--租户的登录 -->
        <filter>
                <filter-name>CAS TenantLogin Filter</filter-name>
                 <filter-class>com.yonyou.iuap.newtenant.cas.TenantSelectFilter</
filter-class>
                <init-param>
                        <param-name>encoding</param-name>
                        <param-value>UTF-8</param-value>
                </init-param>
```

```
                <init-param>
                        <param-name>ignoreUrlPatternType</param-name>
                        <param-value>MUTICONTAINS</param-value>
                </init-param>
                <init-param>
                        <param-name>ignorePattern</param-name>
                        <param-value></param-value>
                </init-param>
        </filter>
    <filter-mapping>
            <filter-name>CAS TenantLogin Filter</filter-name>
            <url-pattern>/*</url-pattern>
     </filter-mapping>
```

2、从 request 中获取登录的租户 id 和该用户可登录的租户 ids

```
AttributePrincipal principal = (AttributePrincipal) request.getUserPrincipal();
principal.getAttributes().get(“tenantId”);
AttributePrincipal principal = (AttributePrincipal) request.getUserPrincipal();
principal.getAttributes().get(“allowTenants”);
```

4.2.4 前后端分离单点登录集成

1、将访问后端接口且需要登录的接口配置单点登录 filter（或租户 filter）进行拦截，Filter 详情查看单点登录接入

2、后端做一个登录接口：login.do（Get 请求），此接口的功能主要是获取用户信息，建立前后端会话 session 后 302（重定向）跳转到前端界面。

```
@RequestMapping("/login")
publicvoiddemoLogin(HttpServletRequestrequest,HttpServletResponse response) {
    //获取登录后的用户信息
    AttributePrincipal principal = (AttributePrincipal)request.getUserPrincipal() ;
    String principal.getName();//登录名
    String userId = (String)principal.get("userId");//用户 id
    //TODO:在此实现前后端的会话 session
    response.sendRedirect("/home/user");//302 跳转到前端页面
}
```

3、前端页面每次打开时候访问一个固定的后端接口判断是否登录，接口请求参数添加 isAjax=1(大小写敏感)，对 ajax 的 success 请求结果进行判断，发现返回的 json 数据里要求登录(json.needrelogin)，则跳转到友户通登录地

址:schema://servername/cas/login?sysid=appsysid&service=urlEncode(步骤 2 中的 login.do 的 URL)，否则继续加载数据，处理示例如下：

```
$.ajax({
            url: apiName,
            dataType: 'json',
            type: method,
            async: isAsync,
            data: options,
            success: function(json){
        if (json &&json.needrelogin) { //对请求结果进行特殊处理
            window.location.href='schema://servername/cas/login?sysid=appsysid&service=urlEncode(步骤 2 中的 login.do 的 URL)';
        }
        if (successCallback&& (typeofsuccessCallback == 'function')) {
            successCallback.call(viewObj, json);
        }
    }.bind(this),
        error: function(xhr, status, err){
        if (errorCallback&& (typeoferrorCallback == 'function')) {
            errorCallback.call(viewObj, xhr, status, err);
        }
    }.bind(this)
});
```

4.2.5 客户端软件登录

客户端(包括移动端和 PC 端)集成友户通登录功能时，可使用友户通提供的 SDK 功能，提供用户输入的账号和密码，获取 accesstoken 即可。具体接口使用可参考根据账号密码获取 accesstoken 及刷新 accesstoken。

4.2.6 客户端注销

客户端登录成功后会获取到 accesstoken，在用户退出登录时，需要销毁取到的 accesstoken，应用自己存储的 accesstoken 由应用的服务端自己处理，CAS Server 端的登录信息需要调用 destroyAccessToken 方法销毁，可参考销毁 accesstoken。

4.2.7 客户端登录后，直接点击 web 页面实现登录

客户端(包括移动端和 PC 端)软件在输入账号密码登录之后，应用对应后端会拿到 accesstoken。此时，如果用户客户端(包括移动端和 PC 端)上集成了 Web 版的链接入口，在用户点击时，可通过 accesstoken 换取一个临时 token,在链接后附加上即可，如：http://idtest.yyuap.com/cas/login?sysid=yht&service=http://idtest.yyuap.com/usercenter&token=tokenvalue ，友户通会根据 token 识别用户登录信息，返回登录后的票据。

具体开发可参考根据 accesstoken 获取临时 token。

4.2.8 修改或重置密码

修改或重置密码分为两种场景：

1、用户在使用过程中由于各种原因，不再想使用原来的密码，此时需要用新的密码替代。
2、用户一段时间未使用，忘记了密码，此时可以选择重置密码，系统在确认用户身份后可以设置新的密码。

可参考修改密码和重置密码的 API 部分。

4.2.9 用户信息获取查询

用户信息获取查询包括两种场景，查询指定用户的详细信息和搜索用户。

1、 查询指定用户
可根据用户登录名或者用户 ID 来查询指定的用户，使用可参考 API 说明部分。

2、 搜索用户
提供根据登录名搜索用户，参考根据登录名模糊查询用户信息。

4.2.10 租户信息获取

租户信息的获取有以下几种使用场景供开发者参考：

1、 登录用户在页面上显示当前用户所在的租户列表，用户可以任意选择切换租户。参考根据用户 id 查询所有的租户。

4.2.11 查询企业信息

实际业务中可能会遇到企业认证的情况，需要把当前所在企业信息查询出来，免去重复输入和错误输入；或者是产品需要给其他企业开发票，需要企业抬头等情况，此时需要获取企业的详细信息。

可以参考根据企业 ID 获取企业信息和根据用户 ID 获取该用户管理的企业信息。

4.2.12 租户切换

用户登录产品后，可能会存在多个租户，而不同租户下显示内容和数据不同；此时用户可以选择切换进行不同租户下操作，那么在切换前首先要做的就是获取到可切换的租户列表，可参考获取产品下可登录的租户列表。其他场景用户可灵活运用。

4.2.13 管理租户下的用户

租户管理员可管理租户下的用户列表，如添加、删除用户、搜索查询等操作，这种场景可以参考分页搜索租户下的用户、批量添加用户、和解除租户与用户间的关联等功能接口。

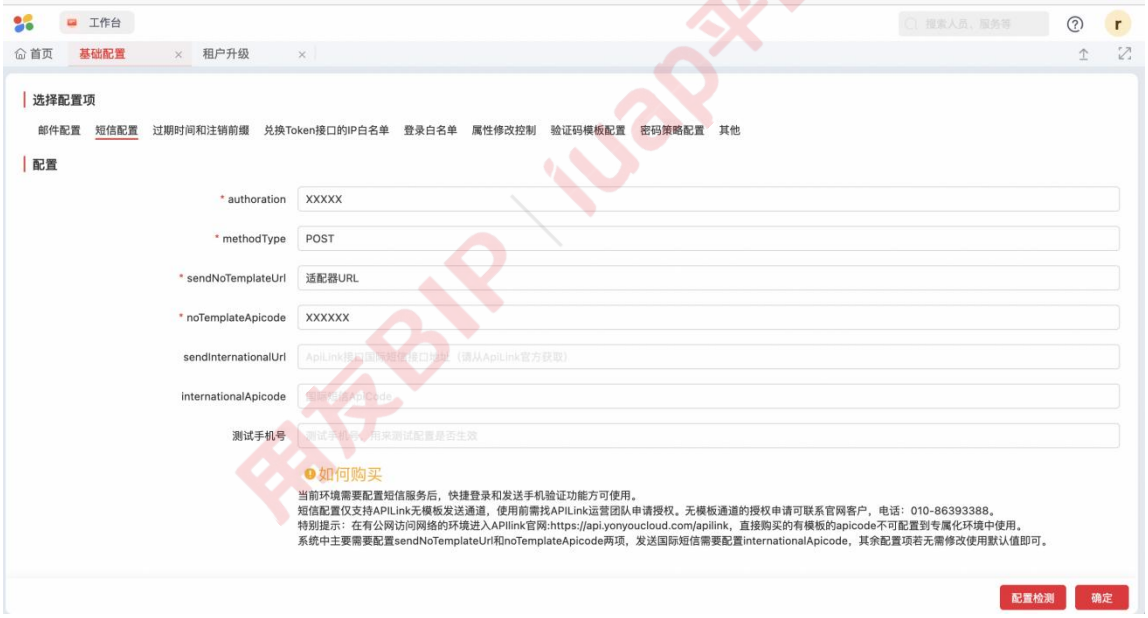
第五章 典型场景客开方案

5.1 专属云-自定义短信通道

5.1.1 场景

客户有自己的短信通道，需要增加一个适配器实现友户通短信调用的接口，然后调用客户短信通道发短信。

在工作台系统管理租户-数字化建模-基础配置的短信配置页签配置自定义的短信服务，sendNoTemplateUrl 参数配置适配器的调用 URL，methodType 固定写 POST，其他参数含义见 3.1.2。



5.1.2 适配器接口以及返回值规范

请求方式：POST

请求参数：

```
{
"msg": "label+msg",      //短信内容
"phone": "手机号"          //接收短信的手机号
}
```

例如：

{

"msg": "【用友网络】验证码：105897，请在页面输入完成验证。",

"phone": "19999999999"

}

Head 里面会存放配置的授权码，key 是上图 authoration 配置的值，value 是 noTemplateApicode 中配置的值，可以简单的对接口进行校验。如果不需要校验，可以忽略。

header：

apicode:xxxxxxxxx

接口的返回结果格式如下：

{

"success": true or false，发送成功返回 true,失败返回 false

"message": "返回信息"

}

第六章 外部系统单点集成方案

6.1 专属云-被集成方案之临时 Token

6.1.1 场景

本方案不适用公有云的友户通单点登录集成。

YonBIP 中的**业务中台**与**技术中台**在用户专属化部署过程中，经常面对用户 SSO 单点登录集成的需求场景。本方案适用于以下涉及专属云友户通的单点登录场景:

1.客户的第三方系统免登访问用友集成了友户通的专属云服务，包括不限于工作台，友报账，智能分析，技术中台，友空间等。

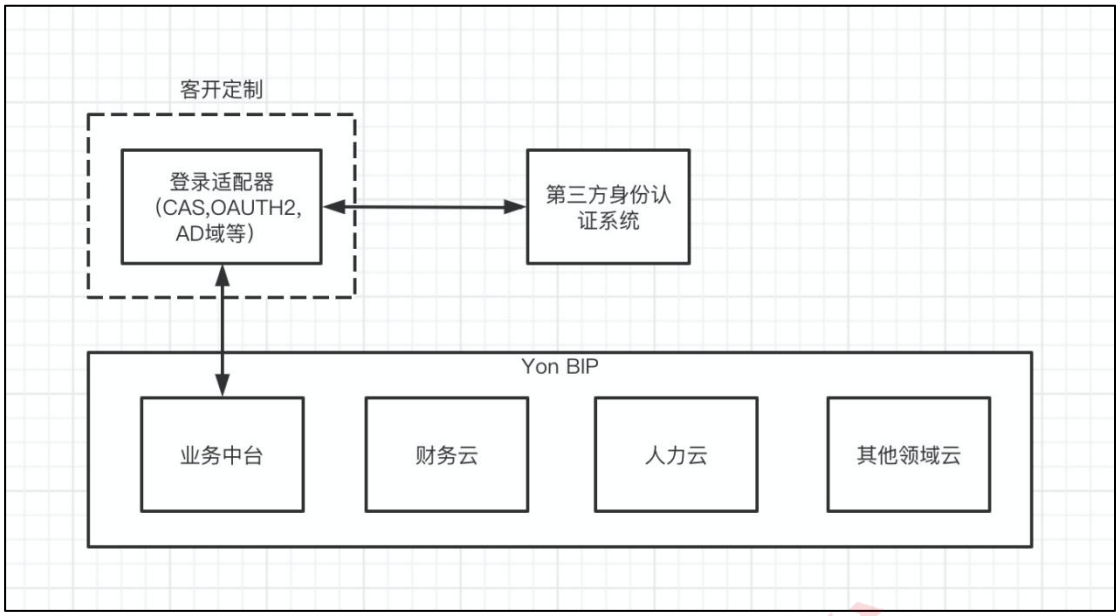
2.用友内部未集成友户通的服务，免登访问集成友户通的专属云服务。

3.客开项目自定义开发登录页面的应用免登访问集成友户通的专属云服务。

友户通是 YonBIP 业务中台和技术中台的登录入口，下面为 YonBIP 中的**业务中台**与**技术中台**在需要集成到用户自有 SSO 单点登录认证服务时，提供解决方案指引。

6.1.2 整体方案

业务中台、技术中台、数据中台已经集成了友户通，友户通作为用友业务中台、技术中台、数据中台的单点登录服务，通过外置第三方登录适配服务，由第三方适配服务与第三方身份认证系统进行集成对接，登录适配器通过统一的方法与友户通进行对接，完成各中台的登录操作。



整体方案的基本过程如下：

1. 业务中台、技术中台、数据中台采用统一的单点登录集成方案集成第三方 sso 系统。
2. 第三方 sso 系统与友户通进行用户的同步及映射。
3. 登录适配器负责与第三方 sso 系统和友户通进行交互，进行互换 token 的操作。

下面详细描述友户通提供被集成能力和相关服务。

6.1.3 通过 userId 获取临时 token

基本信息

描述	根据 userId 获取用户信息和临时登录 token，限制 ip 访问
路径	/cas/exclusive/genLoginTokenByUserIdLimitIp
请求方法	GET

请求参数

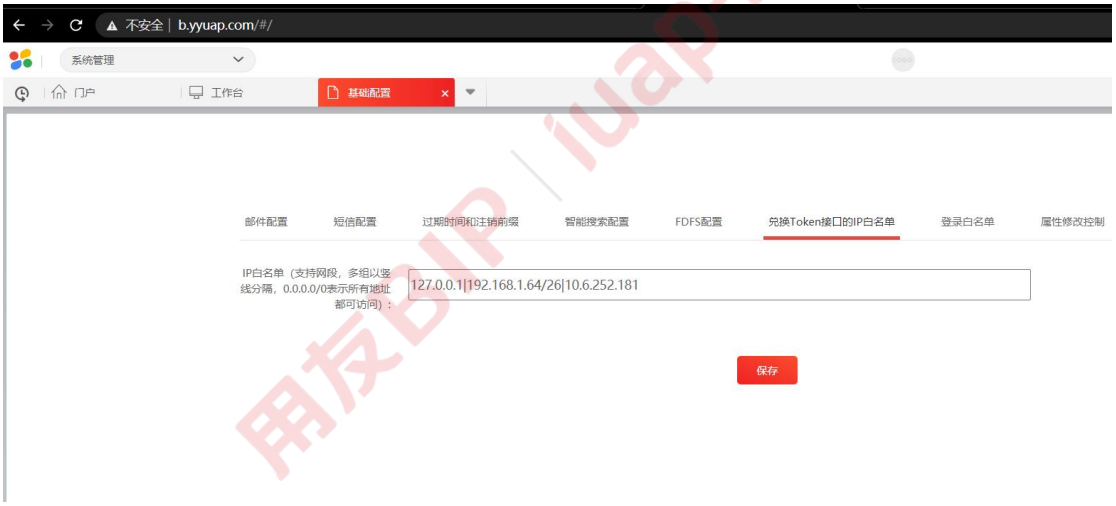
参数名称	类型	描述
userId	String	用户在友户通的 userId

返回参数

参数名称	类型	描述
status	int	1:成功，0:失败
msg	String	提示信息
userId	String	用户的 id
token	String	临时 token

6.1.4 在专属云系统管理租户的基础配置中设置 ip

1.友户通和工作台一起部署：工作台系统管理租户-数字化建模-基础配置



IP 白名单（支持网段，多组以竖线分隔，0.0.0.0/0 表示所有地址都可访问）

6.1.5 使用友户通获取的临时 token 单点到专属云服务

主要作用就是根据 token 可以跳转到登录的服务实现免登，而不用显示登录界面再次登录。token：临时登录 token，5 分钟有效，一次使用。如果前后端分离的 service 地址应该是后端登录接口地址。

地址格式如下：

{域名}/cas/login?token={tokenvalue}&service={service}

地址示例：

http://idtest.yyuap.com/cas/login?sysid=yht&service=http%3A%2F%2Fidtest.yyuap.com%2Fusercenter&token=tokenvalue

注意：service 是要登录后跳转的地址，需要用 urlencode。这个很重要。

Service URL 中包含 tenantInfo 属性是为跳转后再指定当前租户，如不需要指定租户可选择不携带此信息:service=http%3a%2f%2fidtest.yyuap.com%2fusercenter%3ftenantInfo%3du7jv6i73

6.1.6 设置当前租户

设置当前租户根据上节中的{域名}不同有两种情况，一种是专属云工作台地址，一种是友户通 cas 地址。

1、如果是业务中台地址，下面地址需要修改为专属工作台地址，指定租户的参数设置如下：

跳转工作台并指定租户

https://www.diwork.com/login?tenantId=$tenantId

跳转工作台并打开指定服务

https://www.diwork.com/login?tenantId=$tenantId&service=encodeURIComponent(https://www.diwork.com/#/service/$serviceCode)

2、如果是友户通 cas 地址，上面三步完成登录后，再根据第三步 service URL 中的 tenantInfo 属性，Web 端调用设置租户的服务：

基本信息

描述	根据 tenantInfo 设置当前租户
路径	/cas/oauth/setCurrentTenant
请求方法	POST

请求参数

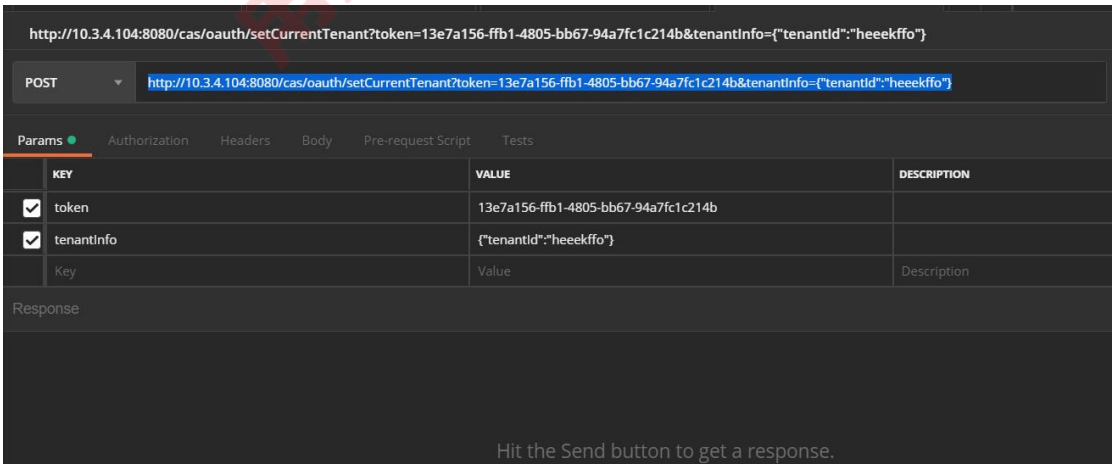
参数名称	类型	描述
tenantInfo	String	租户信息，JSONObject
token	String	第一步获取的临时 token

返回参数

参数名称	类型	描述
status	int	1:成功，0:失败
msg	String	提示信息
userId	String	用户的 id
token	String	临时 token

下面是调用示例：

http:// 域 名 /cas/oauth/setCurrentTenant?token=XXX&tenantInfo={"tenantId":"heeekffo"}



6.1.7 示例代码

代码功能：适配器提供 REST 接口，用于处理拦截的业务中台访问请求。

代码示例 1：

```
@RestController
@RequestMapping("/yht")
public class YhtSSODemo {
    @Autowired
    private static RestTemplate rest;
    @RequestMapping(value = "/testyhtsso", method = RequestMethod.GET)
     public void testyhtsso(HttpServletRequest request, HttpServletResponse response) {
        // 检查是否通过第三方 IDM 认证，如果未认证
        // 调用第三方 IDM 接口登录认证
        // 如果已认证，获取登录用户 ID
        String token = "";
        try {
            // 调用业务中台友互通接口获取登录 TOKEN
            String jsonString = rest.getForObject(
                                                    "http://${ 友 互 通 服 务 地址 }/cas/exclusive/genLoginTokenByUserIdLimitIp?userId=156addad-fa5d-4ee6-8cc7-ffa5e9fba662",
                    String.class);
            System.out.println(jsonString);
            JSONObject obj = JSONValue.parse(jsonString, JSONObject.class);
            token = (String) obj.get("token");
            // 调用业务中台友互通接口，使用获取的 TOKEN 进行校验登录并跳转
                  String workbenchUrl = URLEncoder.encode("${ 业 务 中 台 地址}/workbench/#/", "UTF-8");
            response.sendRedirect(
                    "http://${友互通服务地址}/cas/login?token=" + token + "&service=" + workbenchUrl);
        } catch (Exception e) {
        }
    }

}
```

代码示例 2：

```
import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;
```

```
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.client.RestTemplate;
import net.minidev.json.JSONObject;
import net.minidev.json.JSONValue;

@Controller
@RequestMapping("/yht")
public class YhtSSODemo {

@Autowired
private static RestTemplate rest;

@RequestMapping(value = "/testyhtsso", method = RequestMethod.GET)
public void testyhtsso(HttpServletRequest request, HttpServletResponse response) {
String token = "";
try {
String jsonString = rest.getForObject(
"http://172.20.49.36/cas/exclusive/genLoginTokenByUserIdLimitIp?userId=156addad-fa5d-4ee6-8cc7-ffa5e9fba662",
String.class);
System.out.println(jsonString);
JSONObject obj = JSONValue.parse(jsonString, JSONObject.class);
token = (String) obj.get("token");
response.sendRedirect(
"http://172.20.49.36/cas/login?token=" + token + "&service=http://172.20.49.36/workbench/login/");//service 参数要 urlEncode,若该地址为YonBIP，需要在登录地址后边加/login
} catch (Exception e) {
```

}

}

}

业务中台修改：在 YMS 控制台工作台应用中存在两个配置项，workbench.custom_login_url 对应登录 url，workbench.custom_logout_url 对应登出 url。

6.2 公有云-集成认证中心单点方案

6.2.1 场景

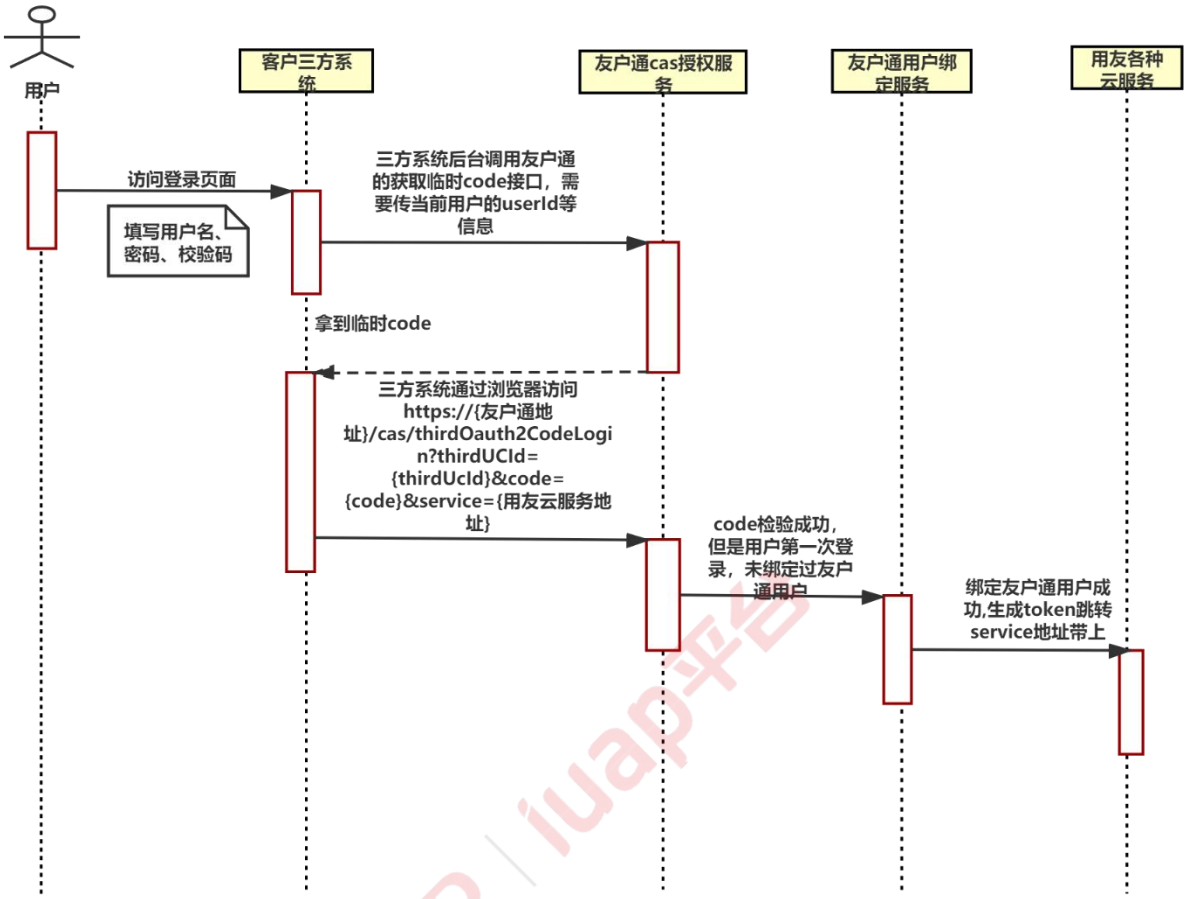
下面以 APP 中集成 H5 页面为例，说明第三方认证单点友户通的实现方案。

现有某房产项目需税务云个人票夹 H5 界面嵌入房产 APP，具体方案大致为：房产 APP 单点友互通，再进入票友记 H5 界面。

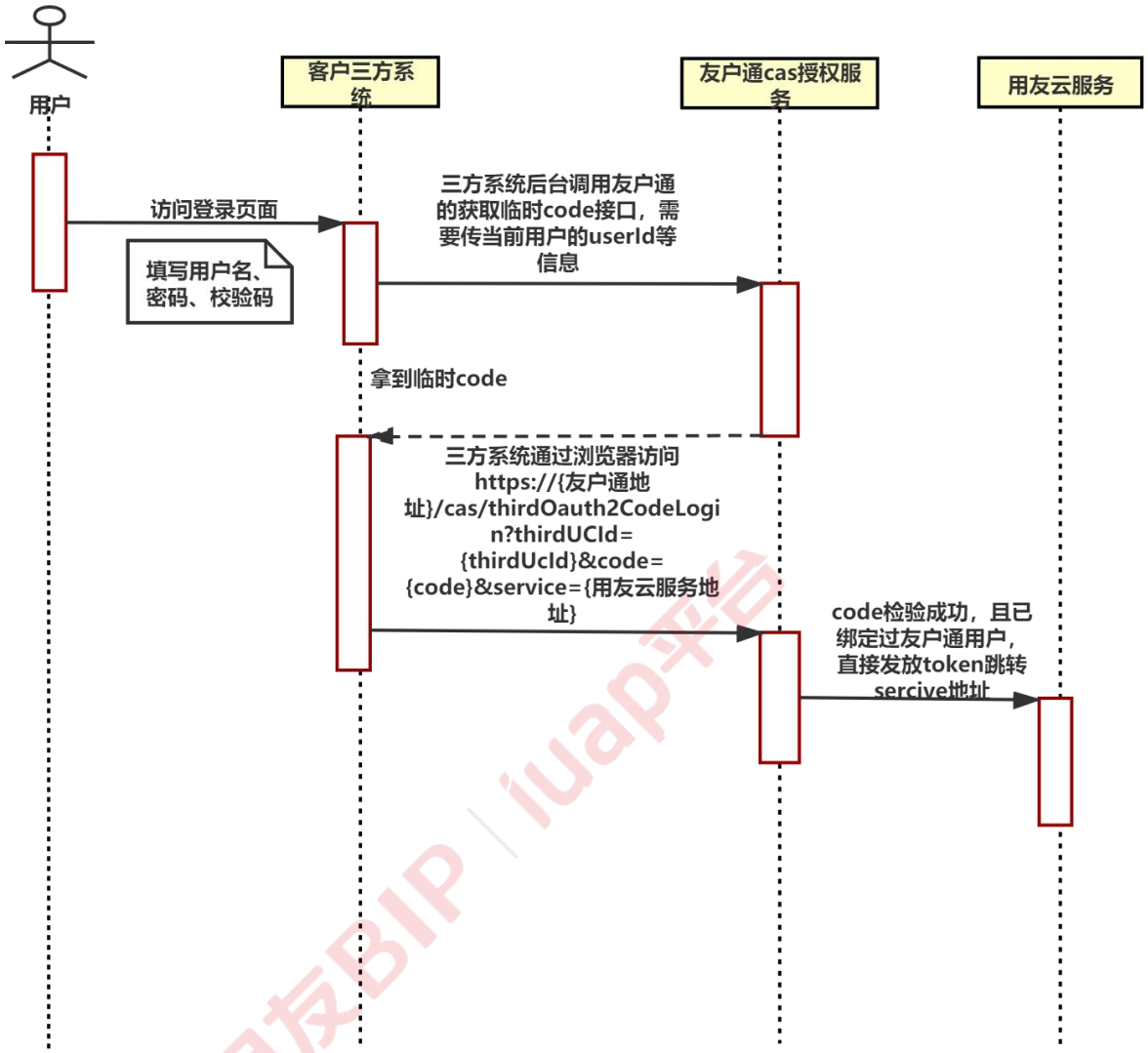
该方案涉及 APP 单点友互通平台，对应的人员需租户管理员在用户管理节节维护对应租户下的用户列表。

6.2.2 整体方案概述

三方集成认证中心用户第一次单点登录用友云服务的时序图：



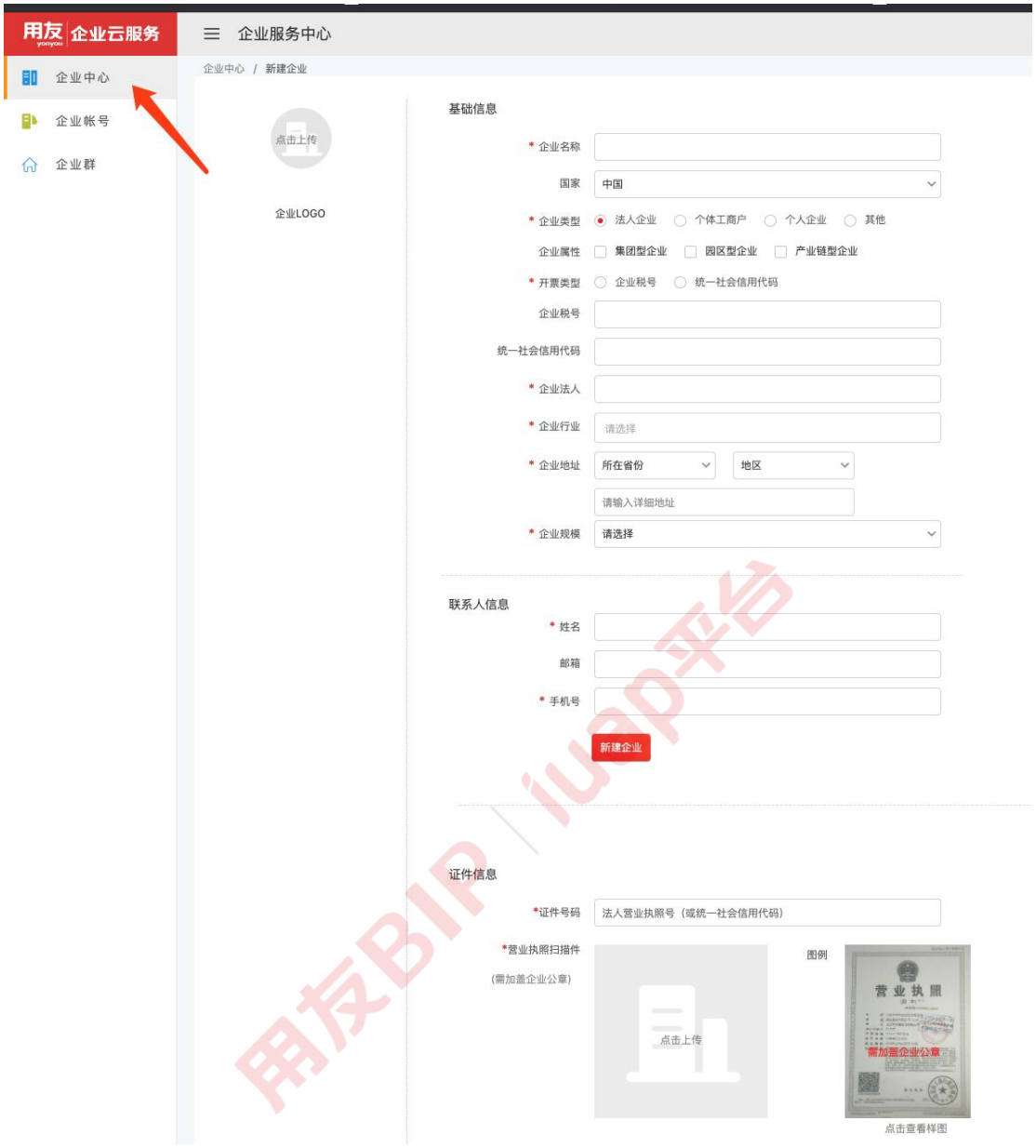
三方集成认证中心用户第二次单点登录用友云服务时序图：



6.2.3 配置集成认证中心

1. 访问数字化建模-系统管理-集成认证中心，点击新增集成认证。（需要当前企业账号先完成企业认证。如未认证可看 2）

2. 如当前企业账号未完成企业认证，可访问企业服务中心，按要求完成企业认证并绑定企业账号。企业服务中心地址：https://apcenter.yonyoucloud.com/apptenant;


用友 企业云服务
企业服务中心
企业中心
企业帐号
企业群
企业中心 / 新建企业
点击上传
企业LOGO
基础信息
* 企业名称
国家
中国
* 企业类型
法人企业
个体工商户
个人企业
其他
企业属性
集团型企业
园区型企业
产业链型企业
* 开票类型
企业税号
统一社会信用代码
企业税号
统一社会信用代码
* 企业法人
* 企业行业
请选择
* 企业地址
所在省份
地区
请输入详细地址
* 企业规模
请选择
联系人信息
* 姓名
邮箱
* 手机号
新建企业
证件信息
*证件号码
法人营业执照号（或统一社会信用代码）
*营业执照扫描件
(需加盖企业公章)
点击上传
图例
营业执照
需加盖企业公章
点击查看样图


用友 企业云服务
企业服务中心
企业中心
企业帐号
企业群
新建企业帐号
致远OA集成测试
未绑定企业
产品
3
用户数
1
diwork

3. 新建集成认证中心后，系统会为当前三方认证中心分配一个编码。

4. 对接开放平台接口，获取单点登录 code，详细接口文档如下：
https://open.diwork.com/#/doc-center/docDes/api?code=diwork&section=d1c38bd20e2e41fbaf200c4fdf883d9b

调用接口需要先完成开放平台 access_token 鉴权。

5. 使用单点 code 拼接登录地址，规则如下
https://euc.diwork.com/cas/thirdOauth2CodeLogin?thirdUCId={thirdUcId}&code={code}&service={service}

参数描述：

thirdUcId	三方认证中心编码
Code	调用开放平台接口返回的 code
Service	登录成功后的跳转地址

service 参数需要进行一次 encode，后端可以采用 URLEncode，前端可以使用 encodeURIComponent

例如：encodeURIComponent('https://www.diwork.com/login')

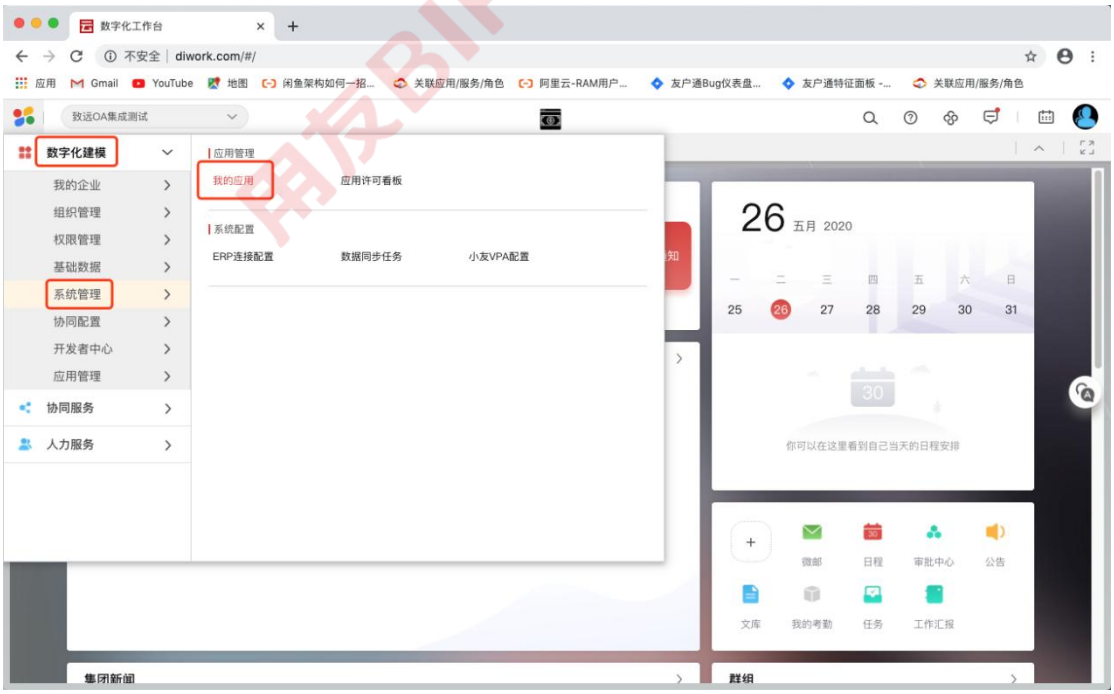
常用 service 举例：

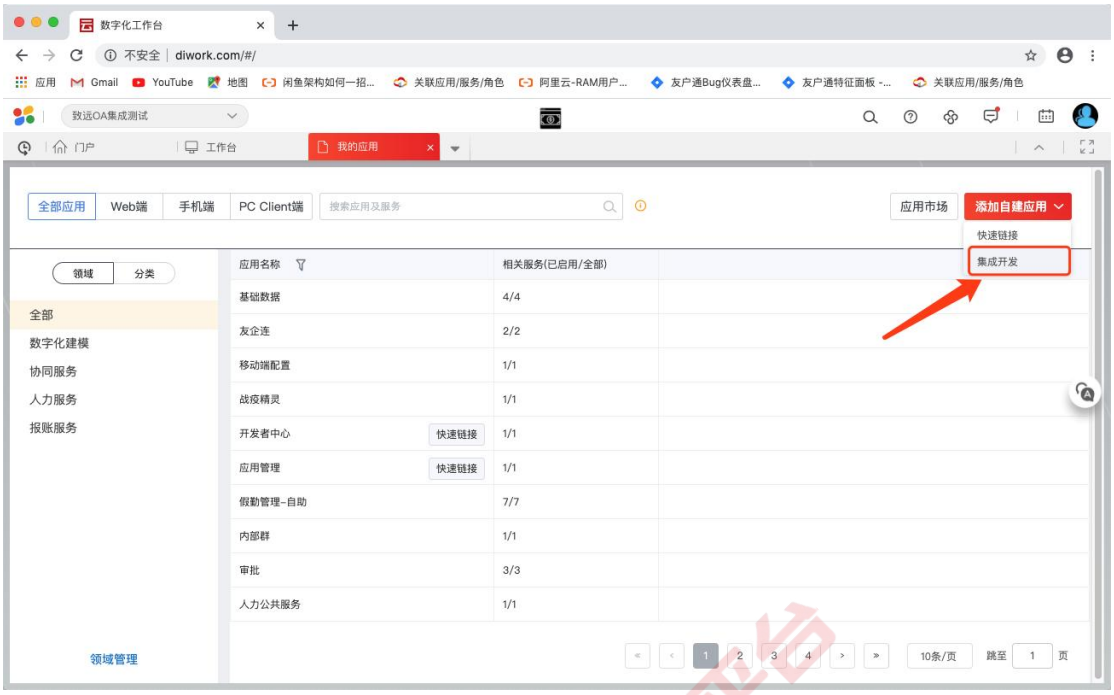
diwork	https://www.diwork.com/login
Yonsuite	https://yonsuite.diwork.com/login

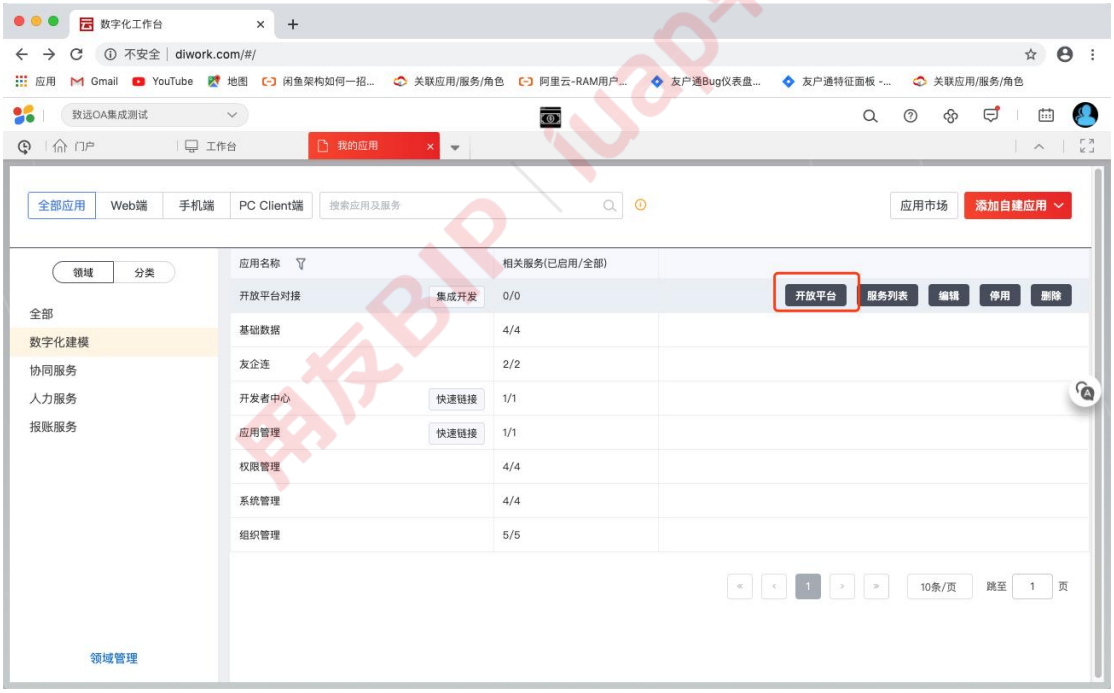
6. 用户访问单点登录地址，首次登陆需要输入用友云账号和验证码，完成账号绑定。已完成绑定的账号可以凭 code 直接登录用友云。用户可在用友云账号中心 https://euc.diwork.com 进行解绑，也可以由管理员在集成认证中心-用户管理里解绑。

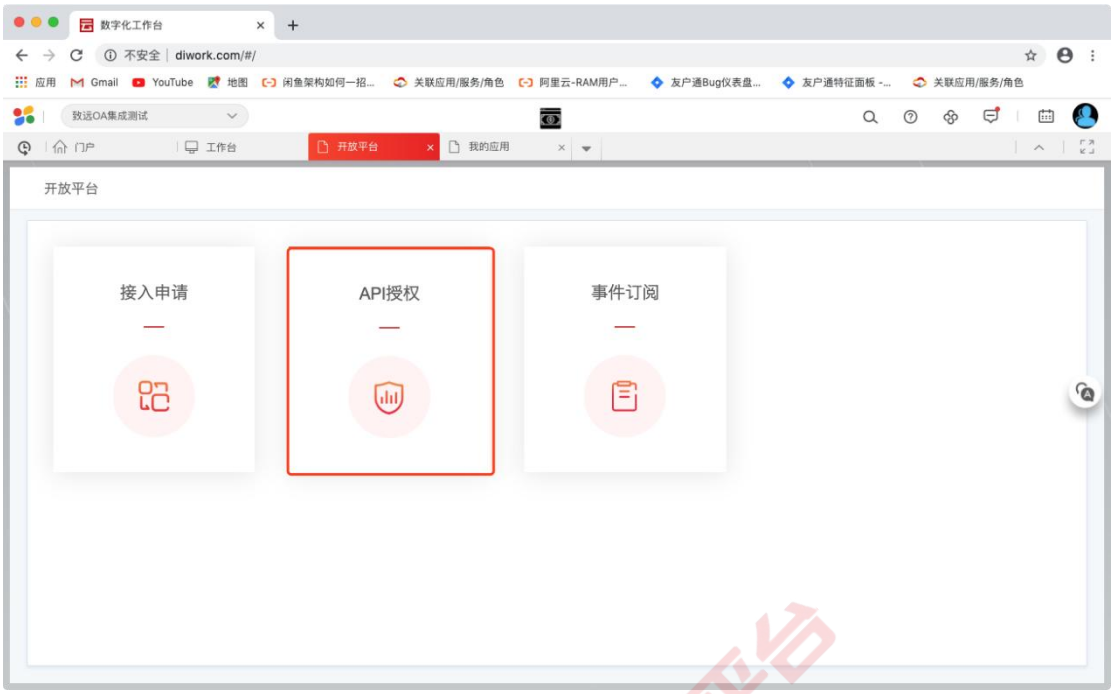
7. 开放平台对接示意：
新建应用 -> 开放平台 AK 获取 -> 接口授权 -> 开发；详细接入过程可参考开放平台-企业自建应用接入文档：

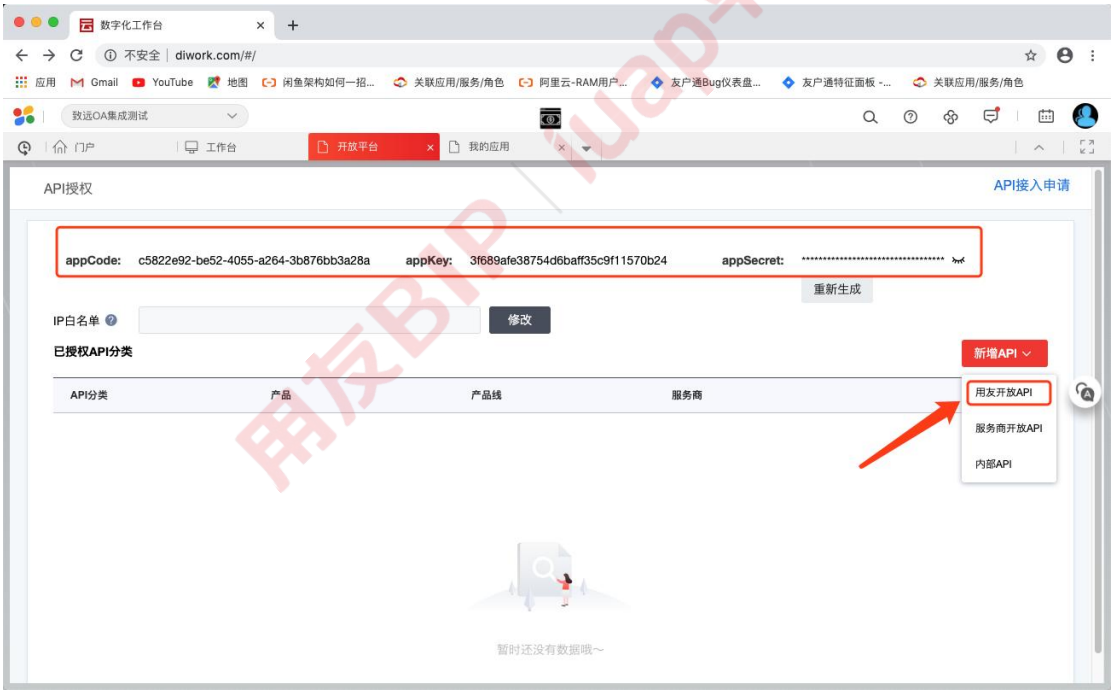
https://open.diwork.com/#/doc-center/docDes/doc?code=open_jrwd&section=022c941650ae4989af7dd6ac7fd4d412

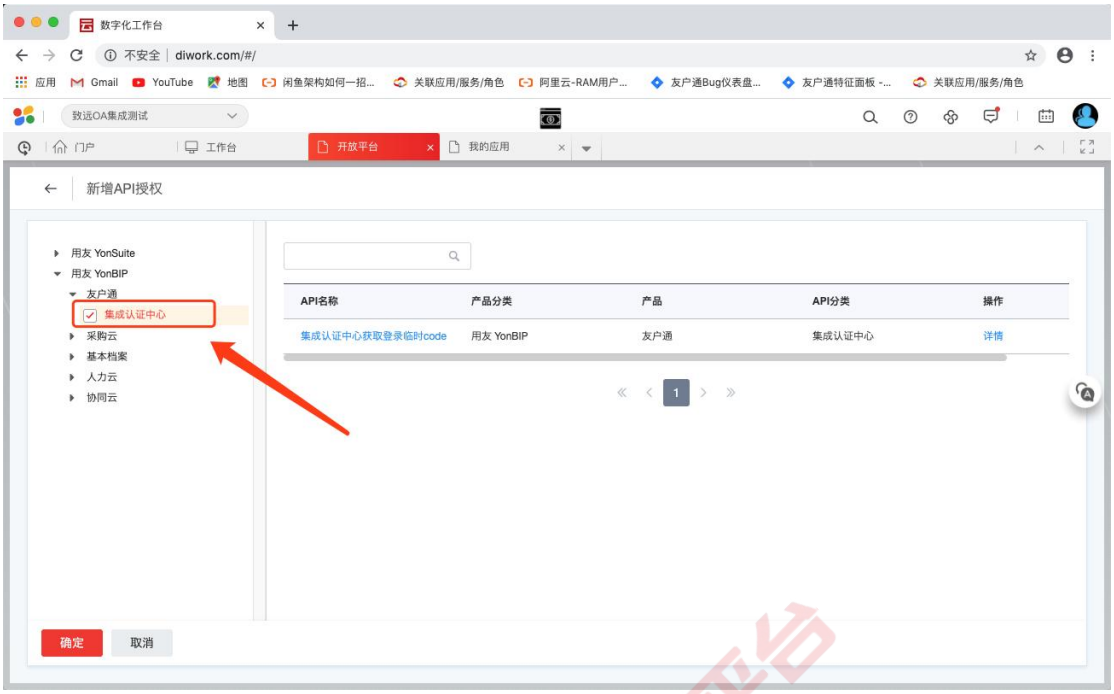







数字化工作台
不安全 | diwork.com/#/
致远OA集成测试
门户
工作台
开放平台
我的应用
开放平台
接入申请
API授权
事件订阅


数字化工作台
不安全 | diwork.com/#/
致远OA集成测试
门户
工作台
开放平台
我的应用
API授权
API接入申请
appCode: c5822e92-be52-4055-a264-3b876bb3a28a
appKey: 3f689afe38754d6baff35c9f11570b24
appSecret:
重新生成
IP白名单
修改
已授权API分类
新增API
API分类
产品
产品线
服务商
用友开放API
服务商开放API
内部API
暂时还没有数据哦~



6.3 AD 域集成

6.3.1 添加 AD 域功能

AD 域属于非标准盘的功能，一般情况下是看不到的，如果客户想要使用，需要在友户通数据库里执行两条 SQL 语句。如下：

```
INSERT INTO iuap_uuas_usercenter.pub_configuration(`key2`, `value2`, `type`, `content`, `type_name`, `data_type`, `description`) VALUES ('ldap_manager_key', '[{"boxValue":[true,false],"isNeed":true,"dataType":"boolean","description":"是否开启 AD 登录","value":true,"key":"switchFlag","inputBox":"radio"},{"isNeed":true,"dataType":"string","name":"AD 域服务的地址","description":"AD 域服务的地址","value":"","key":"host","inputBox":"inputBox"},{"isNeed":true,"dataType":"integer","name":"AD 域服务的端口号","description":"AD 域服务的端口号","value":389,"key":"port","inputBox":"inputBox"},{"isNeed":true,"dataType":"string","name":"AD 域服务的 basedn","description":"AD 域服务的basedn","value":"","key":"basedn","inputBox":"inputBox"},{"isNeed":false,"dataType":"string","name":"用户所在 AD 域的组","description":"用户所在 AD 域的组","key":"domain","inputBox":"inputBox"},{"isNeed":true,"dataType":"string","name":"管理员登录名","description":"管理员登录名","value":"CN=caxxx2010,OU=09Xxxxxce","key":"managerName","inputBox":"inputBox"},{"isNeed":true,"dataType":"string","name":"管理员密码","description":"管理员密码","value":"","key":"managerPass","inputBox":"inputBox"},{"isNeed":true,"dataType":"string","name":"AD 的用户名属性：一般默认cn","description":"AD 的用户名属性：一般默认cn","value":"cn","key":"usercodeKey","inputBox":"inputBox"},{"isNeed":true,"dataType":"string","name":"AD 的实体属性类名，默认*","description":"AD 的实体属性类名，默认*","value":"*","key":"userObject","inputBox":"inputBox"},{"isNeed":true,"dataType":"string","name":"AD 的邮箱属性名，默认mail","description":"AD 的邮箱属性名，默认mail","value":"mail","key":"emailKey","inputBox":"inputBox"},{"boxValue":[true,false],"isNeed":true,"dataType":"boolean","description":"是否开启 SSL 认证","value":false,"key":"sslFlag","inputBox":"radio"},{"isNeed":true,"dataType":"integer","name":"AD 域 SSL 服务的端口号","description":"AD 域 SSL 服务的端口号","value":636,"key":"sslPort","inputBox":"inputBox"}]', 'unify_config', NULL, 'AD 域服务认证', 'json', NULL);
```

```
INSERT INTO iuap_uuas_usercenter.pub_configuration(`key2`, `value2`, `type`, `content`, `type_name`, `data_type`, `description`) VALUES ('union_user_center_key', '[{\"boxValue\":[true,false],\"isNeed\":true,\"dataType\":\"boolean\",\"description\":\"是否开启联邦登录\",\"value\":false,\"key\":\"switchFlag\",\"inputBox\":\"radio\"},{\"isNeed\":true,\"dataType\":\"string\",\"description\":\"公共 domain 地址 \",\"value\":\"yyuap.com\",\"key\":\"domainName\",\"inputBox\":\"inputBox\"},{\"boxValue\":[true,false],\"isNeed\":true,\"dataType\":\"boolean\",\"description\":\"校验是否请求远程服务器 \",\"value\":true,\"key\":\"validateServerFlag\",\"inputBox\":\"radio\"},{\"isNeed\":true,\"dataType\":\"string\",\"description\":\"秘钥 \",\"value\":\"123\",\"key\":\"encryptionKey\",\"inputBox\":\"inputBox\"}]', 'unify_config', NULL, '联邦用户中心', 'json', NULL);
```

6.3.2 配置说明

执行完 SQL 以后，就可以在工作台上看到对应的菜单。使用 yhtmanager 账号登录系统，选择“系统管理”租户，然后在数字化建模下找到基础配置节点，基础配置里就能看到新的菜单【AD 域认证服务】。配置内容说明如下：

是否开启 AD 登录:　是（登录时会先尝试 AD 登录，若登录不成功，会再尝试友户通登录）、否（不会尝试 AD 登录）

AD 域服务的地址:　ip 地址，比如 192.168.8.174

AD 域服务的端口号:　默认 389

AD 域服务的 basedn:　比如 DC=ufsoft,DC=com,DC=cn

用户所在 AD 域的组:　非必填项，限制只能访问 AD 域下的某一个组

管理员登录名:　比如 CN=cams2010,OU=09Service，注意这里不能是短名

管理员密码:　管理员密码（最开始版本为明文，之后会改成加密，在 /yht-user/genpsw 页面可以看到加密工具）

AD 的用户名属性：　一般默认 cn

AD 的实体属性类名：　默认*，修改密码时会用到

AD 的邮箱属性名：　默认 mail，修改密码时会用到

是否开启 SSL 认证:　是（AD 域服务需要开始 SSL 认证，并且开放对应端口）、否

AD 域 SSL 服务的端口号:　默认 636

6.3.3 设置用户默认校验方式为 AD

在【【基础配置】-【密码策略配置】，配置相应的参数

这里只需修改最后两项：

默认密码校验规则:firstAD

默认密码修改规则:firstAD

只有密码策略是 firstAD，那么才会走 AD 登录校验。这里配置好以后，在添加和修改用户的时候，会自动覆盖密码策略。

注意：如果一个用户最开始的密码策略不是 firstAD，这里修改以后也不会马上变成 AD 登录，需要修改数据库里 pub_tenant_user 表的 encode_by 字段。

6.3.4 AD 账号与友户通账号的对应关系

iuap_uuas_usercenter库 pub_tenant_user 表的user_code 字段对应AD域的短名（sAMAccountName）。

6.3.5 密码校验逻辑

标准盘里的功能是：配置过 firstAD 以后，密码校验时先校验 AD 域，如果 AD 域校验不通过，会再次校验友户通的密码。

如果客户有特殊需求，可以联系友户通工作人员，进行定制。