统一认证技术规范

文档版本 V1.0

2021-03-16

修订历史				
编号	修订内容简述	修订日期	修订人	审阅人
1	研发概要设计模版	2021-03-16		
2				
3				

目录

1. 文档目的

本文主要介绍如何通过 PaaS 平台的认证中心，来允许外部系统进行集成，以便实现多个系统的统一认证和单点登录。结合 BFF 及微前端框架，也可以更好的对平台能力进行快速扩展，降低组件之间的耦合性。

2. 技术标准

2.1.Oauth2.0 协议

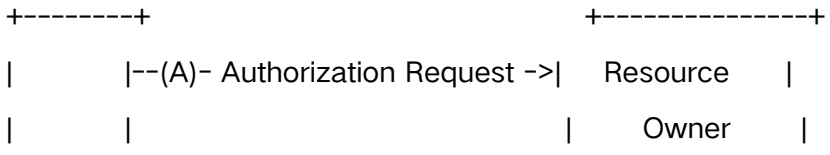
OAuth 是一个开放标准，允许用户让第三方应用访问该用户在某一网站上存储的私密的资源（如照片，视频，联系人列表），而无需将用户名和密码提供给第三方应用。目前，OAuth 的最新版本为 2.0

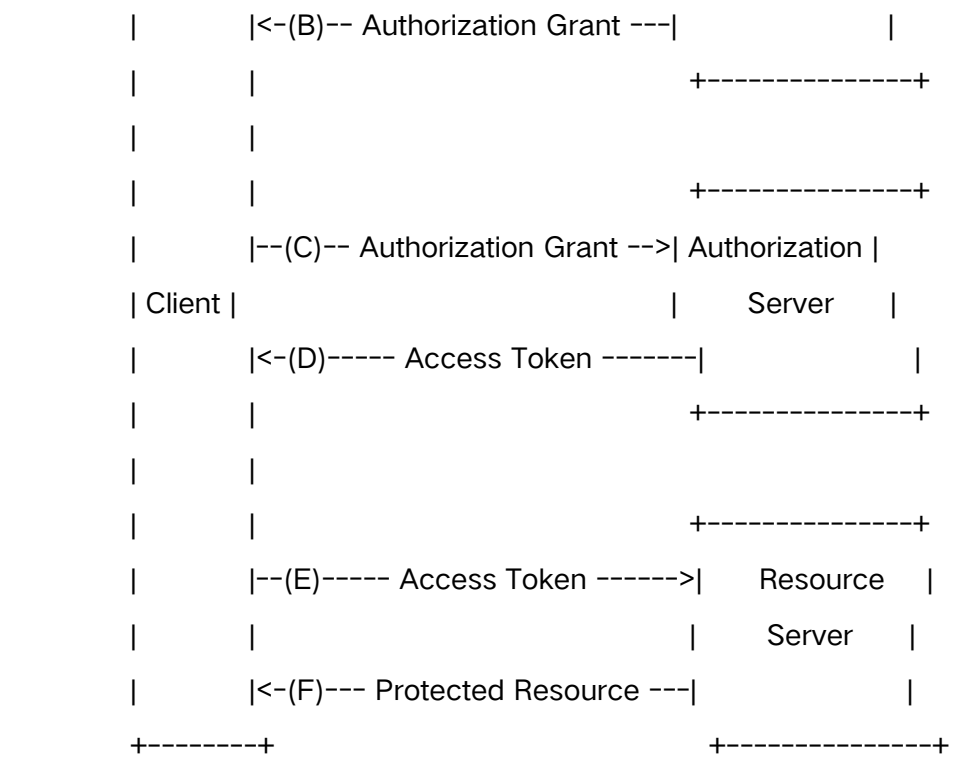
OAuth 允许用户提供一个令牌，而不是用户名和密码来访问他们存放在特定服务提供者的数据。每一个令牌授权一个特定的网站（例如，视频编辑网站)在特定的时段（例如，接下来的 2 小时内）内访问特定的资源（例如仅仅是某一相册中的视频）。这样，OAuth 允许用户授权第三方网站访问他们存储在另外的服务提供者上的信息，而不需要分享他们的访问许可或他们数据的所有内容。

OAuth 2.0 主要有 4 类角色：

- resource owner：资源所有者，指终端的“用户”（user）
- resource server：资源服务器，即服务提供商存放受保护资源。访问这些资源，需要获得访问令牌（access token）。它与认证服务器，可以是同一台服务器，也可以是不同的服务器。如果，我们访问新浪博客网站，那么如果使用新浪博客的账号来登录新浪博客网站，那么新浪博客的资源和新浪博客的认证都是同一家，可以认为是同一个服务器。如果，我们是新浪博客账号去登录了知乎，那么显然知乎的资源和新浪的认证不是一个服务器。
- client：客户端，代表向受保护资源进行资源请求的第三方应用程序。
- authorization server： 授权服务器， 在验证资源所有者并获得授权成功后，将发放访问令牌给客户端。

OAuth 2.0 的认证流程如下：





（A）用户打开客户端以后，客户端请求资源所有者（用户）的授权。

（B）用户同意给予客户端授权。

（C）客户端使用上一步获得的授权，向认证服务器申请访问令牌。

（D）认证服务器对客户端进行认证以后，确认无误，同意发放访问令牌。

（E）客户端使用访问令牌，向资源服务器申请获取资源。

（F）资源服务器确认令牌无误，同意向客户端开放资源。

其中，用户授权有四种模式：

- 授权码模式（authorization code）
- 简化模式（implicit）
- 密码模式（resource owner password credentials）
- 客户端模式（client credentials）

在服务集成场景下，一个服务采用客户端模式（client credentials）来获取其访问令牌（access token），把它作为服务身份标识传递给被调用的其他服务。

客户端模式（client credentials）模式下一个客户端获取授权令牌具体步骤如下：

(A) 客户端向认证服务器进行身份认证，并要求一个访问令牌。请求的参数如下：

- response_type：表示授权类型，必选项，此处的值固定为 "client_credentials"
- client_id：表示客户端的 ID，必选项

•client_secret：客户端的密码，可选项

•scope：表示申请的权限范围，可选项

Http 请求格式如下：

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
grant_type=client_credentials&client_id=s6BhdRkqt3
&client_secret=123456&scope=test
```

(B) 认证服务器确认无误后，向客户端提供访问令牌。

这个过程中，client_id 和 client_secret 是认证服务器分配给一个客户端（client）表明客户端身份的信息。对于一个服务而言，需要小心保存该信息，防止泄露后被恶意服务冒用身份。

2.2.OpenID Connect 1.0 协议

OpenID Connect(目前版本是 1.0)是 OAuth 2.0 协议之上的简单身份层，用 API 进行身份交互的框架，允许客户端根据授权服务器的认证结果最终确认用户的身份，以及获取基本的用户信息；它支持包括 Web、移动、JavaScript 在内的所有客户端类型；它是可扩展的协议，允许你使用某些可选功能，如身份数据加密、OpenID 提供商发现、会话管理等。

✧**OpenID Connect 角色介绍**

1. 客户端：直接为终端用户提供服务
2. 认证服务：OpenID 提供方，通常是一个 OpenID 认证服务器，它能为第三方颁发用于认证的 ID Token
3. 业务服务：提供业务服务，比如查询用户资料
4. 终端用户：指持有资源拥有人

✧**1.2 OpenID Connect 流程描述**

如下图流程所示。

1. 客户端发送认证请求给认证服务；
2. 终端用户在认证页面进行授权确认（可选）；
3. 认证服务对认证请求进行验证，发送 ID Token 给客户端；
4. 客户端向业务请求，请求中携带 ID Token；
5. 业务服务验证 ID Token 是否合法后返回业务应答；

这个流程的第二步是可选的，如果终端用户在客户端输入了用户名和密码，第一步中的认证请求中携带了用户名和密码，那么认证服务在验证了用户名和密码后，省去第二步，可以直接在应答中返 ID Token。这种模式更加简洁。

认证服务返回的 ID Token 需要严格遵守 JWT(JSON Web Token)的定义，下面是 JWT(JSON Web Token)的定义细节：

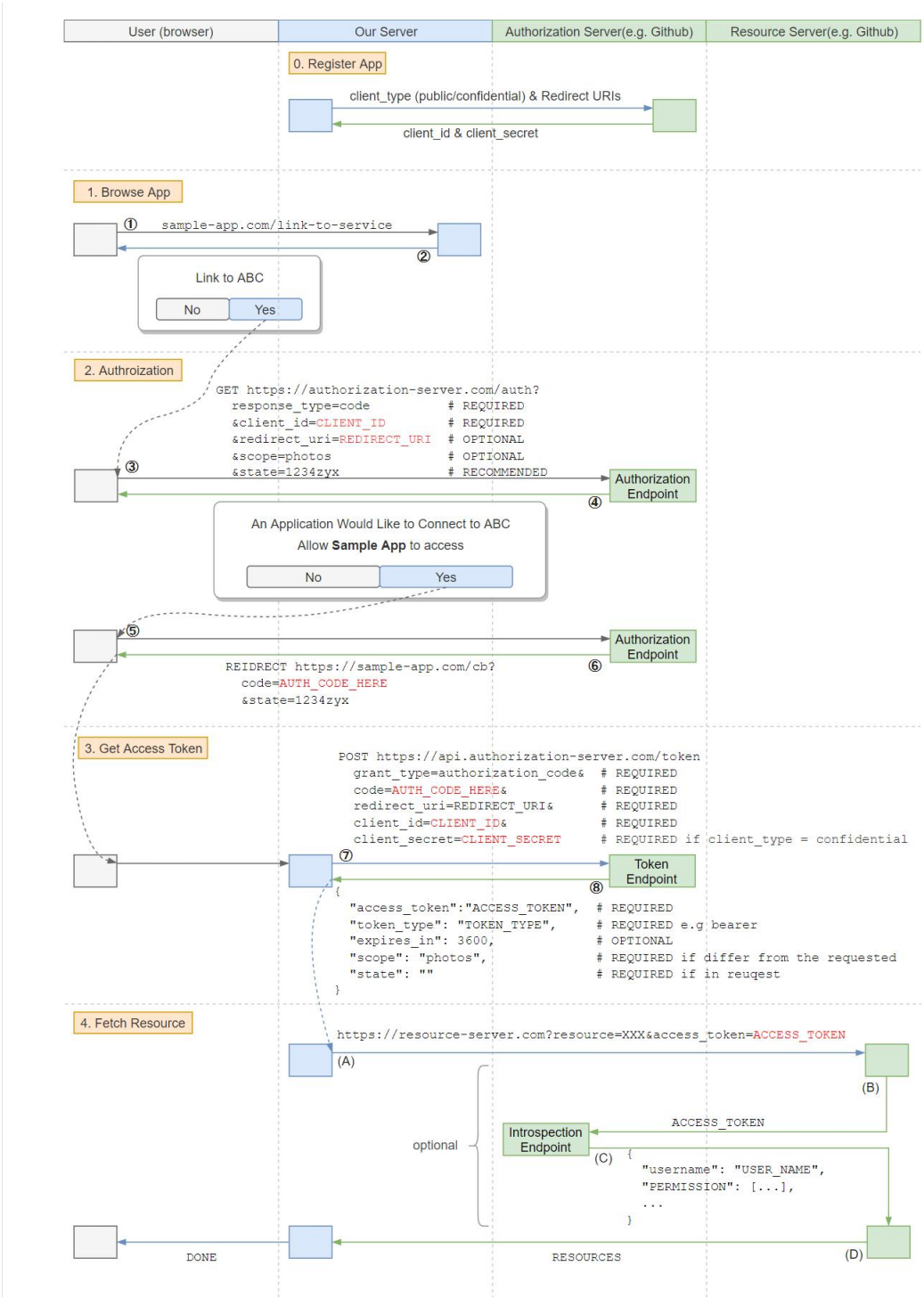
- iss：必须。Issuer Identifier，认证服务的唯一标识，一个区分大小写的 https URL，不包含 query 和 fragment 组件
- sub：必须。Subject Identifier，iss 提供的终端用户的标识，在 iss 范围内唯一，最长为 255 个 ASCII 个字符，区分大小写
- aud：必须。Audience(s)，标识 ID Token 的受众，必须包含 OAuth2 的 client_id，分大小写的字符串数组
- exp：必须。Expiration time，超过此时间的 ID Token 会作废
- iat：必须。Issued At Time，JWT 的构建的时间
- auth_time：AuthenticationTime，终端用户完成认证的时间。
- nonce：发送认证请求的时候提供的随机字符串，用来减缓重放攻击，也可以用来关联客户端 Session。如果 nonce 存在，客户端必须验证 nonce

- acr：可选。Authentication Context Class Reference，表示一个认证上下文引用值，可以用来标识认证上下文类
- amr：可选。Authentication Methods References，表示一组认证方法
- azp：可选。Authorized party，结合 aud 使用。只有在被认证的一方和受众（aud）不一致时才使用此值，一般情况下很少使用

下面是一个典型 ID Token 的示例，供参考：


```
{
    "iss": "https://1.2.3.4:8443/auth/realms/kubernetes",
    "sub": "547cea22-fc8a-4315-bdf2-6c92592a6e7c",
    "aud": "kubernetes",
    "exp": 1525158711,
    "iat": 1525158411,
    "auth_time": 0,
    "nonce": "n-0S6_WzA2Mj",
    "acr": "1",
    "azp": "kubernetes",
    "nbf": 0,
    "typ": "ID",
    "session_state": "150df80e-92a1-4b0c-a5c5-8c858eb5a848",
    "userId": "123456",
    "preferred_username": "theone",
    "given_name": "the",
    "family_name": "one",
    "email": "theone@mycorp.com"
}
```


具体的交互流程参考下图：

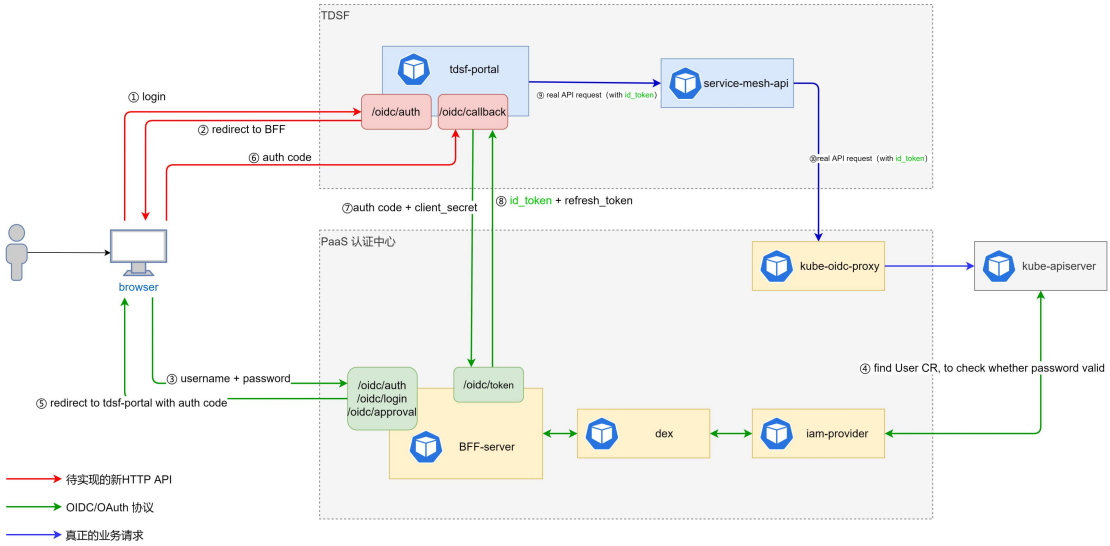


3.集成方式

3.1. 架构及流程

这里介绍一下如何通过平台的统一认证模块，实现同外部系统的集成。平台使用

OIDC 进行 token 的颁发及后续 API 调用的统一认证及鉴权，示例流程如下图所示：



TDSF 是一个示例的外部接入系统，通过 PaaS 的认证中心进行统一认证，认证流程参考前面介绍的 OIDC 协议。

认证后会返回给第三方系统 TDSF 标准的 OIDC token，第三方系统即可通过该 token 对 PaaS 平台的其他 API 进行调用，并访问授权的 API 资源。

3.2. 接入流程示例

1. 环境准备

* 应用地址：http://192.168.2.217:5555
* 认证中心地址：https://192.168.2.216

2. 应用注册

1）应用提供给 TCE：

* 应用名称：如 sample-app
* 回调地址，如：http://192.168.2.217:5555/callback

2）认证中心提供给应用：

* client_id：应用 ID，用于访问认证中心
* client_secret：应用 Secret，用于访问认证中心

3. 用户登录

1）用户访问应用界面，应用跳转到认证中心登陆 URL，URL 示例如下：

```
https://192.168.2.216/auth?client_id=sample-app-id&redirect_uri=http%3A%2F%2F192.168.2.217%3A5555%2Fcallback&response_type=code&scope=openid+profile+email&state=I+wish+to+wash+my+irish+wristwatch
```

参数名称	描述
client_id	应用 ID，注册阶段由 TCE 提供，如：sample-app-id
redirect_uri	应用回调地址，注册阶段由应用提供，如：http://192.168.2.217:5555/callback
response_type	固定值：code
scope	固定值：openid profile email
state	随机字符串，由应用定义，如：3rd party application to integrate

2）认证中心：弹出登陆页面，用户输入用户名/密码登陆（如果用户已经登陆，这步会自动跳过）

3）认证中心：跳转回应用注册的回调地址，跳转示例如下：

```
http://192.168.2.217:5555/callback?code=kf7dmmvhdipdcjczydklwi6pu&state=3rd+party+application+to+integrate
```

参数名称	描述
code	授权码，由 TCE 认证服务 生成，用于后面获取 token
state	与应用跳转到 TCE 认证服务登陆 URL 时传递的 state 值一样

4）应用后台调用 TCE token API 获取 token 信息，调用示例如下：

```
curl -XPOST 'https://192.168.2.216/token' \
-H 'Authorization: Basic c2FtcGxlLWFwcC0yMTg6WlhoaGJYQnNaUzFoY0hBdGMyVmpjbVYwJw==' \
-H 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'code=kf7dmmvhdipdcjczydklwi6pu' \
--data-urlencode 'grant_type=authorization_code' \
```

```
--data-urlencode 'redirect_uri=http://192.168.2.217:5555/callback'
```

参数名称	描述
Authorization	格式 Basic XXXX，其中 XXXX 是 client_id:client_secret 的 base64 编码
Content-Type	固定值：application/x-www-form-urlencoded
code	授权码
grant_type	固定值：authorization_code
redirect_uri	应用回调地址，TCE 认证服务会验证该地址是否与应用注册的回调地址一致

5）返回 token 信息示例

```
ID Token:
eyJhbGciOiJSUzI1NiIsImtpZCI6ImQxY2Y2MzA3YmQ5Yzk3OTJiMzdmMGJiM2M1Njk1ZDQ0MWJlZTMzNjcifQ.eyJpc3MiOiJodHRwczovLzE5Mi4xNjguMi4yMTYiLCJzdWIiOiJDZ0V4RWdsMFpXNTRZMnh2ZFdRliwiYXVkIjoic2FtcGxlLWFwcC0yMTgiLCJleHAiOjE2MzAwMzA0ODEsImlhdCI6MTYyOTk0NDA4MSwiYXRfaGFzaCI6Ik1PUjk0enktTUZNcU5zZUZTM1ZzRXciLCJjX2hhc2giOiJpbkoteDVKUEFCRXhaaEpRaEx3T3pBliwiZW1haWwiOiJhZG1pbkBleGFtcGxlLmNvbSIsImVtYWlsX3ZlcmlmaWVkIjp0cnVlLCJuYW1lIjoiYWRtaW4iLCJwaG9uZSI6IjE3MzQzMTM1MDUxliwidXNlcmlkIjoiMSJ9.FrC6oKRsManuP9opqugknJmOE78uKmxX6uteM1flCDVRqRv-riG0C5AOX4K9BTnT9Gllu3H24jydT4ybSissz_wL_mLzoTQWoQ9uMMmd4w1aiGqgO6mIaEh3XvTqtoQv1ltONSkp49bykpdlXbDJxy0PScU0k-0XFNJIMSBwn8SEubgH7NO3xwFzsjaLqBfolxC5YXBuWS8n-FEOqNTg-mx-n_Fu2oemJCT-8qWMqY6FNjRSC3D-2ABkCbl4g76vPLgJ-l6dU6eaJvaBW6S4BzhCX0SitxYrxcXjOGviX1HKOXXSUC1n1HfQpOpNW-FA2G3F-kON94rYr1AEdlwSVw

Access Token:
eyJhbGciOiJSUzI1NiIsImtpZCI6ImQxY2Y2MzA3YmQ5Yzk3OTJiMzdmMGJiM2M1Njk1ZDQ0MWJlZTMzNjcifQ.eyJpc3MiOiJodHRwczovLzE5Mi4xNjguMi4yMTYiLCJzdWIiOiJDZ0V4RWdsMFpXNTRZMnh2ZFdRliwiYXVkIjoic2FtcGxlLWFwcC0yMTgiLCJleHAiOjE2MzAwMzA0ODEsImlhdCI6MTYyOTk0NDA4MSwiYXRfaGFzaCI6IlhYS3RzUkhZS043WnZGOUFxcXVSd3ciLCJlbWFpbCI6ImFkbWluQGV4YW1wbGUuY29tliwiZW1haWxfdmVyaWZpZWQiOnRydWUsIm5hbWUiOiJhZG1pbiIsInBob25lIjoiMTczNDMxMzUwNTEiLCJ1c2VyaWQiOiIxIn0.VXUxba9cy4S3ZUtyfbF6A3Xg5fZYL-m6nqR09qiJgJmNUCEkHr_b4xKXXvDJwsgoC0zdUxShd1BpOgN4vi8A7zZ676-PybB2dVT6x3EPpwCVC8_NVYwXijeELWJJ0nU9aAq6p_m-XdXOuPzutbmMLSWVfrY-CS2WSdQlSuKxb28slTptzCF4OY3dZugOHZ_v10KAxsMo0Aul4d3C_EXOaiUKzw0OTD7xyYOm8MmJvhzQEYSodHAkbJWnOusKEBtFKJ5hhAqPLCymY2VleN-7Jbqr-DYrDSDtd7FF1vCbDL0-rTwQ5_79FlAu_fusOcVc26M7GlvOtGcniaweNtm1dg
```

返回的 ID Token 是标准 JWT Token，应用可以从 ID Token 中解析出用户信息，例如：

```
{
    "iss": "https://192.168.2.216",
    "sub": "CgExEgl0ZW54Y2xvdWQ",
    "aud": "sample-app",
    "exp": 1630030481,
    "iat": 1629944081,
    "at_hash": "MOR94zy-MFMqNseFS3VsEw",
    "c_hash": "inJ-x5JPABExZhJQhLwOzA",
```

```
    "email": "admin@example.com",
    "email_verified": true,
    "name": "admin",
    "phone": "17343135051",
    "userid": "1"
}
```


Access Token 可以通过认证中心获取用户详细信息，或者调用其他授权的资源。

4. 用户登出

应用通过认证中心的登出 URL 进行注销，如：https://192.168.2.216/oidc/logout。