ALIBABA CLOUD

阿里云

访问控制

开放授权管理（OAuth）

文档版本：20220519

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.OAuth应用概览 05
2.OAuth应用典型场景 07
2.1. Web应用登录阿里云 07
2.2. Native应用登录阿里云 12
2.3. 通过OIDC获取用户信息 17
2.4. 通过SCIM协议将企业内部账号同步到阿里云RAM 23
3.管理OAuth应用 29
3.1. 创建应用 29
3.2. 查看应用基本信息 29
3.3. 修改应用基本信息 29
3.4. 添加应用范围 30
3.5. 创建应用密钥 30
3.6. 删除应用 30
4.OAuth常用的SDK示例 32

1.OAuth应用概览

RAM支持使用OAuth 2.0协议进行用户认证和应用授权。本文介绍OAuth 2.0的基本概念和应用场景。

OAuth基本概念

概念	说明
用户	用户需要登录到阿里云并授权应用访问阿里云资源。用户可以是阿里云账号（主账号），也可以是RAM用户。
阿里云OAuth 2.0服务	对用户进行认证，并接受用户对应用的授权，生成代表用户身份的令牌并返回给被授权的应用。
OAuth应用	获取用户授权，并获取代表用户身份的令牌，从而可以访问阿里云。OAuth 2.0服务目前支持的应用类型包括： • **WebApp**：指基于浏览器交互的网络应用。 • **NativeApp**：指操作系统中运行的本地应用，主要为运行在桌面操作系统或移动操作系统中的应用。 • **ServerApp**：指直接访问阿里云服务，而无需依赖用户登录的应用。目前仅支持基于SCIM协议的用户同步应用。
OAuth范围	OAuth 2.0服务通过OAuth范围来限定应用扮演用户登录阿里云后可以访问的范围。支持的范围如下： • *openid*：用于获取用户的OpenID（默认权限范围，不可移除）。 **说明** 获取到的OpenID是一个可以唯一代表用户的字符串，但并不包含阿里云UID、用户名等信息。如果需要这些信息，需要使用aliuid、profile等权限范围。 • *aliuid*：用于获取阿里云UID。 • *profile*：用于获取用户名称等个人信息。 • */acs/ccc*：用于访问阿里云呼叫中心服务API。 • */acs/alidns*：用于访问阿里云解析API。 • */acs/scim*：用于访问阿里云跨域身份管理服务。 • */acs/digitalstore*：用于访问数字化门店（Digital Store）。 • */acs/scsp*：用于访问智能客服平台（Smart Customer Service Platform）。 • */acs/cloudgame*：用于访问云游戏平台（Cloud Gaming Platform）。 • */acs/aiccs*：用于访问智能联络中心。
令牌	OAuth 2.0服务可以给应用下发代表登录用户的令牌。 • 身份令牌：包含用户的身份信息，不能用于访问阿里云资源。 • 访问令牌：包含用户的身份信息以及应用的OAuth范围，可以用于访问OAuth范围内的阿里云资源。 • 刷新令牌：用于换取新的访问令牌。

概念	说明
阿里云API	应用通过调用API可以访问相应资源。

OAuth的应用场景

- Web应用登录阿里云
- Native应用登录阿里云
- 通过OIDC获取用户信息
- 通过SCIM协议将企业内部账号同步到阿里云RAM

2.OAuth应用典型场景

2.1. Web应用登录阿里云

本文介绍Web应用如何通过OAuth 2.0扮演登录用户访问阿里云API。

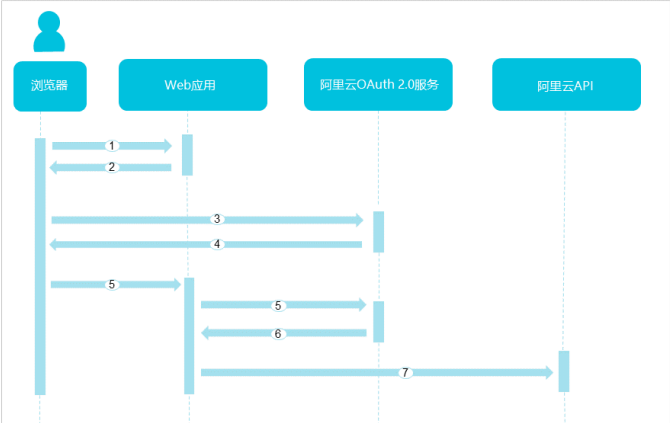
前提条件

- Web应用扮演登录用户访问阿里云首先需要创建应用，为应用提供恰当的名称、OAuth范围、回调地址等关键信息，详情请参见创建应用。应用创建成功后，您可以在应用列表中查看应用ID（client_id），如下图所示。

> **说明** 应用创建成功之后，可以在阿里云账号内直接扮演用户。如果要扮演其他阿里云账号的用户，需要获得其他阿里云账号的授权。

- 创建应用密钥，详情请参见创建应用密钥。应用密钥（client_secret）只在创建时显示，不支持查询，请妥善保管。

基本流程



1. 用户通过浏览器登录Web应用。
2. Web应用重定向到阿里云OAuth 2.0服务并将URL返回给浏览器。

   > **说明** 如果用户还未登录，则会进一步重定向到阿里云登录服务。

3. 用户通过浏览器登录阿里云OAuth 2.0服务并申请授权码。
4. 阿里云OAuth 2.0服务重定向到Web应用并返回授权码给浏览器。
5. 浏览器通过Web应用使用授权码向阿里云OAuth 2.0服务申请代表用户身份的令牌。
   - 如何获取访问令牌，请参见获取访问令牌。

- 如何获取新的访问令牌，请参见获取新的访问令牌。
- 如何撤销刷新令牌，请参见撤销刷新令牌。

6. 阿里云OAuth 2.0服务向Web应用返回访问令牌（access_token）。
7. Web应用通过获取的访问令牌（access_token）向阿里云发起访问API的请求。

> **说明** 由于访问令牌（access_token）可以代表用户身份，因此应用可以访问当前用户的资源。

获取访问令牌

1. Web应用通过浏览器将用户重定向到阿里云OAuth 2.0服务从而获取授权码。

   授权码的请求地址：`https://signin.aliyun.com/oauth2/v1/auth`。

   请求参数

参数名称	是否必选	描述
client_id	是	应用ID。
redirect_uri	是	创建应用的重定向URI之一。
response_type	是	返回类型。根据OAuth 2.0协议，目前支持设置此参数的取值为code。
scope	否	空格分隔的OAuth范围列表。如不指定此参数取值，则默认为应用的全部OAuth范围。
access_type	否	应用的访问类型。取值分为两种类型： ○ *online*：应用不需要离线刷新访问令牌。 ○ *offline*：针对离线访问类型的请求，会发放刷新令牌，应用可以根据需求持续刷新访问令牌。 默认取值为*online*。
state	否	应用通过state 参数实现多种目的，例如：状态保持、作为nonce使用从而减少CSRF威胁等。state如果设置为任意字符串，阿里云OAuth2.0服务会将请求中的state参数及取值原样放到返回参数中以供后续使用。
prompt	否	该参数用于指定服务器是否需要提示用户进行授权操作。 如果指定，会强制要求用户进行授权，即使该阿里云账号已经做过授权，也需要重新授权。如果不指定，则只在该阿里云账号第一次使用该应用时要求授权。 取值：`admin_consent`，表示服务端会在向客户端返回信息之前展示授权页面。

请求示例

```
https://signin.aliyun.com/oauth2/v1/auth?
client_id=123****
redirect_uri=https://example.com/authcallback/&
response_type=code&
scope=openid /acs/ccc&
access_type=offline&
state=123456****
```

返回示例

```
GET HTTP/1.1 302 Found
Location: https://example.com/authcallback/?code=ABAFDGDFXYZW888&state=123456****
```

2. Web应用使用授权码向阿里云OAuth 2.0服务申请代表用户身份的令牌。

换取访问令牌的请求地址：`https://oauth.aliyun.com/v1/token`。

请求参数

参数名称	是否必选	描述
code	是	初始请求中获取的授权码。
client_id	是	应用ID。
redirect_uri	是	初始请求中设置的参数。
grant_type	是	根据OAuth 2.0协议， 取值为*authorization_code*。
client_secret	否	应用密钥，用作换取访问令牌时鉴定应用身份的密码。

请求示例

```
POST /v1/token HTTP/1.1
Host: oauth.aliyun.com
Content-Type: application/x-www-form-urlencoded
code=ABAFDGDFXYZW888&
client_id=123****
client_secret=`your_client_secret`&
redirect_uri=https://example.com/authcallback/&
grant_type=authorization_code
```

返回参数

参数名称	描述

参数名称	描述
access_token	访问令牌。 访问令牌可以代表用户身份，应用使用此访问令牌来访问阿里云API。应用不需要理解访问令牌的含义，直接使用即可。
expires_in	访问令牌的剩余有效时间，单位为秒。
token_type	访问令牌的类型。取值为*Bearer*。
id_token	身份令牌。 身份令牌为OAuth签名的JWT（JSON Web Token）。如果初始请求的scope参数包含了openid，则返回身份令牌。
refresh_token	刷新令牌。 如果初始请求时应用的访问类型为offline，则返回刷新令牌。
scope	获取的授权码的实际授权范围。 如果应用需要的授权范围不止 `openid` ，那么您需要校验获取的scope是否符合预期。 如果您安装的应用未授权指定的scope，请您重新授权。重新授权的方法：通过 `https://signin.aliyun.com/oauth2/v1/auth` 请求授权码时，设置参数 `prompt=admin_consent` 。

返回示例

```
{
  "access_token": "eyJraWQiOiJrMTIzNCIsImVu****",
  "token_type": "Bearer",
  "expires_in": "3600",
  "refresh_token": "Ccx63VVeTn2dxV7ovXXfLtAqLLERA****",
  "id_token": "eyJhbGciOiJIUzI1****",
  "scope": "openid /acs/ccc"
}
```

获取新的访问令牌

换取访问令牌的请求地址： `https://oauth.aliyun.com/v1/token` 。

请求参数

参数名称	是否必选	描述
refresh_token	是	用授权码换取访问令牌时获得的刷新令牌。
client_id	是	应用ID。
grant_type	是	根据OAuth 2.0协议， 取值为*refresh_token*。
client_secret	否	应用密钥，用作换取访问令牌时鉴定应用身份的密码。

请求示例

```
POST /v1/token HTTP/1.1
Host: oauth.aliyun.com
Content-Type: application/x-www-form-urlencoded
refresh_token=Ccx63VVeTn2dxV7ovXXfLtAqLLERAH1Bc&
client_id=123****
client_secret=`your_client_secret`&
grant_type=refresh_token
```

返回参数

参数名称	描述
access_token	新的访问令牌。应用可以使用新的访问令牌来访问阿里云API。
expires_in	访问令牌的剩余有效时间，单位为秒。
token_type	访问令牌的类型。取值为*Bearer*。

返回示例

```
{
  "access_token": "eyJraWQiOiJrMTIzNCIsImVu****",
  "token_type": "Bearer",
  "expires_in": "3600"
}
```

撤销刷新令牌

Web应用获取了刷新令牌后，在用户退出登录应用或用户将自己的账号从应用中移除时，必须撤销刷新令牌。

撤销刷新令牌的请求地址： `https://oauth.aliyun.com/v1/revoke` 。

请求参数

参数名称	是否必选	描述
token	是	需要撤销的刷新令牌。
client_id	是	应用ID。
client_secret	否	应用密钥。

2.2. Native应用登录阿里云

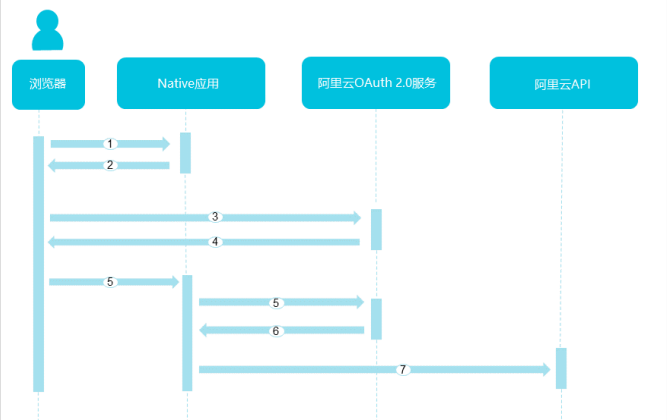
本文介绍桌面和移动端的Native应用如何通过OAuth 2.0扮演登录用户访问阿里云API。

前提条件

Native应用扮演登录用户访问阿里云首先需要创建应用，为应用提供恰当的名称、OAuth范围、回调地址等关键信息。详情请参见创建应用。由于Native应用运行在非可信环境，无法有效保护应用密钥，因此Native应用不使用应用密钥。

> **说明** 应用创建成功之后，可以在云账号内直接扮演用户。

基本流程



1. 用户通过浏览器登录Native应用。
2. Native应用重定向到阿里云OAuth 2.0服务并将URL返回给浏览器。

   > **说明** 如果用户还未登录，则会进一步重定向到阿里云登录服务。

3. 用户通过浏览器登录阿里云OAuth 2.0服务并申请授权码。
4. 阿里云OAuth 2.0服务重定向到Native应用并返回授权码给浏览器。
5. 浏览器通过Native应用使用授权码向阿里云OAuth 2.0服务申请代表用户身份的令牌。
   - 如何获取访问令牌，请参见获取访问令牌。
   - 如何获取新的访问令牌，请参见获取新的访问令牌。
   - 如何撤销刷新令牌，请参见撤销刷新令牌。
6. 阿里云OAuth 2.0服务向Native应用返回令牌。
7. Native应用通过获取的令牌向阿里云发起访问API的请求。

> **说明** 由于令牌可以代表用户身份，因此应用可以访问当前用户的资源。

Proof Key机制的原理

Native应用支持Proof Key机制，用于每次获取授权码以及用授权码换取访问令牌。

> **说明** 这一机制可以减轻针对授权码截获的攻击。

1. Native应用生成：`code_verifier`，并保存好这个随机字符串。

   > **说明** `code_verifier` 是一个高熵值的随机字符串，其取值为：`[A-Z] / [a-z] / [0-9] / "-" / "." / "_" / "~"`，长度限制为：43~128个字符。

2. 应用根据 `transform` 的方式选择生成 `code_challenge`。应用在申请授权码的同时提交：`code_challenge` 以及生成 `code_challenge` 的方式。

   ```
   code_challenge = transform(code_verifier, [Plain|S256])
   ```

方式	取值
`plain`	如果 `transform` 的方式选择为 `plain`，那么 `code_challenge` 与 `code_verifier` 的值相同。
`S256`	如果 `transform` 的方式选择为 `S256`，那么 `code_challenge` 等于 `code verifier` 的SHA256哈希值。 `code_challenge=BASE64URL-Encode(SHA256(ASCII(code_verifier)))` **说明** 哈希算法的输入为 `code_verifier` 的ASCII编码串，哈希算法的输出字符串需要进行BASE64-URL编码。

   `code_challenge` 选择方式计算示例：

   如果应用采用方式为S256，生成 `code_verifier` 的值为*dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk*，那么 `code_challenge` 为*E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM*。

3. 应用获取授权码后，在使用授权码换取访问令牌时，服务端通过计算判断是否颁发令牌。

   授权码需包括 `code_verifier`，服务端按照应用选择的 `transform` 方式对 `code_verifier` 进行计算，将结果与 `code_challenge` 进行对比，如果一致则颁发访问令牌。

获取访问令牌

1. Native应用通过浏览器将用户重定向到阿里云OAuth 2.0服务从而获取授权码。

   授权码的请求地址：`https://signin.aliyun.com/oauth2/v1/auth`。

请求参数

参数名称	是否必选	描述
client_id	是	应用的身份ID。
redirect_uri	是	创建应用的重定向URI之一。
response_type	是	返回类型。根据OAuth 2.0协议，目前支持设置此参数的取值为*code*。
scope	否	空格分隔的OAuth范围列表。如不指定此参数取值，则默认为应用的全部OAuth范围。
state	否	应用通过state参数实现多种目的，例如：状态保持、作为nonce使用从而减少CSRF威胁等。state如果设置为任意字符串，阿里云OAuth 2.0服务会将请求中的state参数以及取值原样放到返回参数中以供后续使用。
code_challenge_method	否	如果不指定此参数，则默认取值为*plain*。
code_challenge	否	根据客户端指定的code_challenge_method，对随机生成的code_verifier进行转换和编码，转换的结果作为code_challenge参数的值在授权码请求中使用。 **说明** 如果不指定此参数，则无法使用Proof Key机制，在使用授权码换取令牌时也不需要指定相应的code_verifier，若授权码被截获可以直接被用于换取访问令牌。

请求示例

```
https://signin.aliyun.com/oauth2/v1/authorize?
client_id=98989****
&redirect_uri=meeting%3A%2F%2Fauthorize%2F
&response_type=code
&scope=openid%20%2Fworksuite%2Fuseraccess
&state=123456****
&code_challenge=E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSst****
&code_challenge_method=S256
```

返回示例

```
GET HTTP/1.1 302 Found
Location: meeting://authorize/?code=ABAFDGDFXYZW888&state=123456****
```

2. Native应用使用授权码向阿里云OAuth 2.0服务申请代表用户身份的令牌。

   换取访问令牌的请求地址：`https://oauth.aliyun.com/v1/token`。

   请求参数

参数名称	是否必选	描述
code	是	初始请求中获取的授权码。
client_id	是	应用的身份ID。
redirect_uri	是	初始请求中设置的参数。
grant_type	是	根据OAuth 2.0协议， 取值为*authorization_code*。
code_verifier	否	初始请求中生成的code verifier，对应于授权码请求中的code_challenge参数。

   请求示例

```
POST /v1/token HTTP/1.1
Host: oauth.aliyun.com
Content-Type: application/x-www-form-urlencoded
code=ABAFDGDFXYZW888&
client_id=98989****
redirect_uri=meeting://authorize/&
grant_type=authorization_code&
code_verifier=dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
```

   返回参数

参数名称	描述
access_token	访问令牌。访问令牌可以代表用户身份，应用使用此访问令牌来访问阿里云API。应用不需要理解访问令牌的含义，直接使用即可。
expires_in	访问令牌的剩余有效时间，单位为秒。
token_type	访问令牌的类型。取值为*Bearer*。
id_token	身份令牌。id_token为OAuth签名的JWT（JSON Web Token）。如果初始请求的scope参数包含了*openid*，则返回身份令牌。
refresh_token	刷新令牌。此参数可以直接使用，Native应用不需要指定访问令牌的值，可以直接获得刷新令牌。

   返回示例

```
{
  "access_token": "eyJraWQiOiJrMTIzNCIsImVuYyI6****",
  "token_type": "Bearer",
  "expires_in": 3600,
  "refresh_token": "Ccx63VVeTn2dxV7ovXXfLtAqLLERA****",
  "id_token": "eyJhbGciOiJIUzI1****"
}
```

获取新的访问令牌

换取访问令牌的请求地址：`https://oauth.aliyun.com/v1/token`。

请求参数

参数名称	是否必选	描述
refresh_token	是	用授权码换取访问令牌时获得的刷新令牌。
client_id	是	应用的身份ID。
grant_type	是	根据OAuth 2.0协议，取值为*refresh_token*。

请求示例

```
POST /v1/token HTTP/1.1
Host: oauth.aliyun.com
Content-Type: application/x-www-form-urlencoded
refresh_token=Ccx63VVeTn2dxV7ovXXfLtAqLLERAH****
client_id=98989****
grant_type=refresh_token
```

返回参数

参数名称	描述
access_token	新的访问令牌。应用使用新的访问令牌来访问阿里云API。
expires_in	访问令牌的剩余有效时间，单位为秒。
token_type	访问令牌的类型。取值为*Bearer*。

返回示例

```
{
  "access_token": "eyJraWQiOiJrMTIzNCIsImVuYyI6****",
  "token_type": "Bearer",
  "expires_in": 3600,
}
```

> **说明** 本次请求的返回值与用授权码换取访问令牌的返回值一致，但不包含refresh_token和id_token。

撤销刷新令牌

Native应用获取了刷新令牌后，在用户退出登录应用时或用户将自己的账号从应用中移除时，必须撤销刷新令牌。

撤销刷新令牌的请求地址：`https://oauth.aliyun.com/v1/revoke`。

请求参数

参数名称	是否必选	描述
token	是	需要撤销的刷新令牌。
client_id	是	应用的身份ID。

2.3. 通过OIDC获取用户信息

OIDC（OpenID Connect）是建立在OAuth 2.0基础上的一个认证协议，本文为您介绍应用如何使用OIDC获取阿里云登录用户的信息。

前提条件

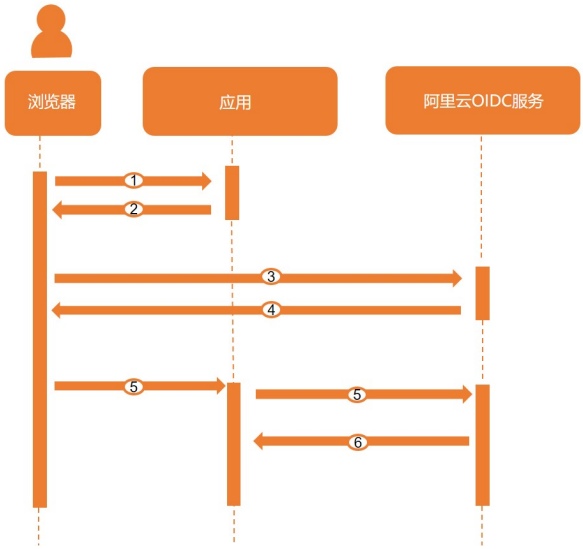
获取用户登录信息前，您需要创建应用，设置应用名称、OAuth范围和回调地址等关键信息，并为应用生成应用密钥。具体操作，请参见创建应用、添加应用范围和创建应用密钥。

基本概念

概念	说明
身份令牌	OIDC可以给应用下发代表登录用户的身份令牌。身份令牌用于获取姓名、登录名等用户信息，不能用于访问阿里云服务。

概念	说明
OIDC Discovery Endpoint	OIDC协议包含了不同的Endpoint用于不同的目的，Discovery Endpoint中包含OIDC协议所需要的所有配置信息，方便开发者使用。 **说明** Discovery Endpoint是通过JSON文档来提供一系列键值，其中包含主要的提供者信息，例如：协议支持的响应类型、令牌颁发者的取值、身份令牌签名密钥的地址和签名算法等。 阿里云作为OIDC服务提供者，提供了一个Discovery Endpoint：`https://oauth.aliyun.com/.well-known/openid-configuration` 来简化配置流程。 Discovery Endpoint包含的内容示例如下： `{ "code_challenge_methods_supported": ["plain", "S256"], "subject_types_supported": ["public"], "response_types_supported": ["code"], "issuer": "https://oauth.aliyun.com", "jwks_uri": "https://oauth.aliyun.com/v1/keys", "revocation_endpoint": "https://oauth.aliyun.com/v1/revoke", "token_endpoint": "https://oauth.aliyun.com/v1/token", "id_token_signing_alg_values_supported": ["RS256"], "scopes_supported": ["openid", "aliuid", "profile"], "authorization_endpoint": "https://signin.aliyun.com/oauth2/v1/auth" }`

基本流程



1. 用户通过浏览器登录应用。
2. 应用重定向到阿里云OIDC服务并将URL返回给浏览器。

   说明 如果用户还未登录，则会进一步重定向到阿里云登录服务。

3. 用户通过浏览器登录阿里云OIDC服务并申请授权码。
4. 阿里云OIDC服务重定向到应用并返回授权码给浏览器。
5. 浏览器通过应用使用授权码向阿里云OIDC服务申请身份令牌。
6. 阿里云OIDC服务向应用返回身份令牌和访问令牌，应用通过身份令牌或访问令牌便可以获取用户信息。

   具体的使用场景如下：

   - 直接解析身份令牌获取用户信息。

     该场景需要应用验证身份令牌的签名。应用首先获取身份令牌签名密钥，然后验证身份令牌的JWT签名，最后使用解析后的用户信息。更多信息，请参见示例一：应用获取身份令牌签名密钥、示例二：验证身份令牌的JWT签名和示例三：解析身份令牌获取用户信息。

   - 使用身份令牌在各个不同模块之间通信。

     该场景需要应用验证身份令牌的签名。应用首先获取身份令牌签名密钥，然后验证身份令牌的JWT签名。更多信息，请参见示例一：应用获取身份令牌签名密钥和示例二：验证身份令牌的JWT签名。

   - 使用访问令牌多次查询用户信息。

     应用在获取访问令牌后，通过调用UserInfo接口获取用户信息。更多信息，请参见示例四：通过访问令牌和UserInfo接口获取用户信息。

示例一：应用获取身份令牌签名密钥

请求示例如下：

```
private List getSignPublicKey() {
  HttpResponse response = HttpClientUtils.doGet("https://oauth.aliyun.com/v1/keys");
  List rsaKeyList = new ArrayList();
  if (response.getCode() == 200 && response.isSuccess()) {
    String keys = JSON.parseObject(response.getData()).getString("keys");
    try {
      JSONArray publicKeyList = JSON.parseArray(keys);
      for (Object object : publicKeyList) {
        RSAKey rsaKey = RSAKey.parse(JSONObject.toJSONString(object));
        rsaKeyList.add(rsaKey);
      }
      return rsaKeyList;
    } catch (Exception e) {
      LOG.info(e.getMessage());
    }
  }
  LOG.info("GetSignPublicKey failed:{}", response.getData());
  throw new AuthenticationException(response.getData());
}
```

示例二：验证身份令牌的JWT签名

阿里云颁发的身份令牌是带有签名的JWT（JSON Web Token），签名算法为JWS标准RS256。当应用请求获取用户信息时，阿里云需要对身份令牌进行验证，包含以下几个方面：

- 签名验证：通过OAuth服务公布的签名公钥，验证身份令牌的真实性和完整性。

  请求示例如下：

```
public boolean verifySign(SignedJWT signedJWT) {
  List publicKeyList = getSignPublicKey();
  RSAKey rsaKey = null;
  for (RSAKey key : publicKeyList) {
    if (signedJWT.getHeader().getKeyID().equals(key.getKeyID())) {
      rsaKey = key;
    }
  }
  if (rsaKey != null) {
    try {
      RSASSAVerifier verifier = new RSASSAVerifier(rsaKey.toRSAPublicKey());
      if (signedJWT.verify(verifier)) {
        return true;
      }
    } catch (Exception e) {
      LOG.info("Verify exception:{}", e.getMessage());
    }
  }
  throw new AuthenticationException("Can't verify signature for id token");
}
```

- 有效期验证：检查令牌颁发时间和令牌过期时间的有效性。
- 检查令牌接收者：防止颁发给其他应用的身份令牌被传递给本应用。

示例三：解析身份令牌获取用户信息

- 返回参数
  - Header返回参数

参数名称	描述	需要的OAuth范围
alg	签名算法。	*openid*
kid	验证身份令牌签名使用的公钥，用户需要使用此公钥验证签名，防止身份令牌被篡改。	*openid*

  - Body返回参数

参数名称	描述	需要的OAuth范围
exp	令牌过期时间戳。	*openid*
sub	唯一代表登录用户的字符串，但并不包含阿里云UID、用户名等信息。	*openid*
aud	令牌接收者，OAuth应用ID。	*openid*
iss	令牌颁发者。取值为*https://oauth.aliyun.com*。	*openid*
iat	令牌颁发时间戳。	*openid*
name	登录用户的显示名称。	*profile*
upn	RAM用户的登录名称。RAM用户请求时才会返回该参数。	*profile*
login_name	阿里云账号（主账号）的登录名称。阿里云账号请求时才会返回该参数。	*profile*
aid	登录用户所属的阿里云账号ID。	*aliuid*
uid	登录用户的ID。如果登录的是阿里云账号（主账号），则此值为阿里云账号ID（与aid值相同），如果登录的是RAM用户，则此值为RAM用户ID。	*aliuid*

- 返回示例
  - Header返回示例

```
{
  "alg": "RS256",
  "kid": "JC9wxzrhqJ0gtaCEt2QLUfevEUIwltFhui4O1bh****"
}
```

- Body返回示例

  为了阅读方便，以下展示的是未编码的身份令牌的返回示例。实际返回为编码后的身份令牌，更多信息，请参见示例二：验证身份令牌的JWT签名。

  - 阿里云账号请求时的Body返回示例

```
{
  "exp": 1517539523,
  "sub": "123456789012****",
  "aud": "4567890123456****",
  "iss": "https://oauth.aliyun.com", //中国站
  "iat": 1517535923,
  "name": "alice", //登录用户的显示名称
  "login_name":"alice@example.com", //阿里云账号的登录名称
  "aid": "123456789012****", //阿里云账号ID
  "uid": "123456789012****" //阿里云账号ID
}
```

  - RAM用户请求时的Body返回示例

```
{
  "exp": 1517539523,
  "sub": "123456789012****",
  "aud": "4567890123456****",
  "iss": "https://oauth.aliyun.com",
  "iat": 1517535923,
  "name": "alice", //登录用户的显示名称
  "upn": "alice@example.onaliyun.com", //RAM用户的登录名称
  "aid": "123456789012****", //RAM用户所属的阿里云账号ID
  "uid": "234567890123****" //RAM用户ID
}
```

示例四：通过访问令牌和UserInfo接口获取用户信息

除了直接获取身份令牌，您也可以在获取访问令牌后通过调用UserInfo接口获取用户信息，该接口必须使用访问令牌才能访问，返回信息不编码。

> **说明** OIDC场景下，即只有*openid*、*aliuid*和*profile*这几个范围的时候，也会返回访问令牌，此时的访问令牌只能用于调用UserInfo接口。

UserInfo接口请求地址：`https://oauth.aliyun.com/v1/userinfo`。

请求示例如下：

```
GET v1/userinfo HTTP/1.1
Host: oauth.aliyun.com
Authorization: Bearer SlAV32hkKG
```

返回参数如下表所示：

参数名称	描述	需要的OAuth范围

参数名称	描述	需要的OAuth范围
sub	唯一代表登录用户的字符串，但并不包含阿里云UID、用户名等信息。	*openid*
name	登录用户的显示名称。	*profile*
upn	RAM用户的登录名称。RAM用户请求时才会返回该参数。	*profile*
login_name	阿里云账号（主账号）的登录名称。阿里云账号请求时才会返回该参数。	*profile*
aid	登录用户所属的阿里云账号ID。	*aliuid*
uid	登录用户的ID。如果登录的是阿里云账号（主账号），则此值为阿里云账号ID（与aid值相同），如果登录的是RAM用户，则此值为RAM用户ID。	*aliuid*

Body返回示例如下：

- 阿里云账号请求时的Body返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "sub": "123456789012****",
  "name": "alice", //登录用户的显示名称
  "login_name":"alice@example.com", //阿里云账号的登录名称
  "aid": "123456789012****", //阿里云账号ID
  "uid": "123456789012****" //阿里云账号ID
}
```

- RAM用户请求时的Body返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "sub": "123456789012****",
  "name": "alice", //登录用户的显示名称
  "upn": "alice@example.onaliyun.com", //RAM用户的登录名称
  "aid": "123456789012****", //RAM用户所属的阿里云账号ID
  "uid": "234567890123****" //RAM用户ID
}
```

2.4. 通过SCIM协议将企业内部账号同步到阿里云RAM

企业上云的过程中，为减少维护和管理成本，需要基于一套原则将企业内部系统中已有的账号同步到云上。阿里云通过SCIM标准协议，结合OAuth应用的安全授权，可以将企业内部系统中的账号数据同步到阿里云访问控制（RAM）中。

前提条件

- 您的阿里云账号或RAM用户有创建OAuth应用的权限。
- 您的阿里云账号或RAM用户有为ServerApp授权的权限，且您仅能给自己账号下的ServerApp授权。

背景信息

- SCIM（System for Cross-domain Identity Management）主要用于多用户的云应用身份管理。SCIM 2.0建立在一个对象模型上，所有SCIM对象都继承Resource。Resource具有id、externalId和meta属性，RFC7643定义了扩展公共属性的User、Group和EnterpriseUser。本文的示例将采用User来同步用户信息。更多关于SCIM的信息，请参见SCIM。
- 开放授权OAuth（Open Authorization）是为用户资源的授权定义了一个安全、开放及简单的标准，第三方无需知道用户的账号和密码，就可获取到用户的授权信息。阿里云当前的OAuth授权形态分为**WebApp**、**NativeApp**和**ServerApp**三种。**WebApp**和**NativeApp**都属于3-legged OAuth，而**ServerApp**属于2-legged OAuth。SCIM主要使用2-legged OAuth机制来完成应用程序（消费者）和API（服务提供商）之间的授权，从而进行下一步的数据同步业务。更多关于OAuth的信息，请参见OAuth 2.0和OAuth应用概览。
- 阿里云SCIM Endpoint：`https://scim.aliyun.com`。
- 阿里云OAuth获取access_token的Endpoint：`https://oauth.aliyun.com/v1/token`。

步骤一：创建OAuth应用并授权

1. 使用阿里云账号登录RAM控制台。
2. 在左侧导航栏，单击**OAuth应用管理**。
3. 在**企业应用**页签，单击**创建应用**，创建OAuth应用。

   其中，**应用类型**选择**ServerApp**。具体操作，请参见创建应用。
4. 单击步骤创建的应用名称，在**应用OAuth范围**页签，单击**添加OAuth范围**，添加OAuth应用范围。

   其中，**OAuth范围**选择**/acs/scim**。具体操作，请参见添加应用范围。
5. 为OAuth应用授权。
   i. 在**应用OAuth范围**页签，单击**去授权**。
   ii. 在**应用授权**页面，选中**阿里云跨域身份管理服务**，然后单击**授权**。
6. 在**应用密钥**页签，单击**创建密钥**，创建应用密钥。

> **注意**
> - 应用密钥内容（client_secret）仅在创建时可见，请您及时保存。
> - **AppSecretId**是应用密钥的ID，不是应用密钥内容（client_secret）。

步骤二：同步账号数据

支持通过客户端（例如：One Identity）或SCIM API两种方式同步账号数据。

- 客户端：您可以在客户端（One Identity）中配置SCIM信息，然后通过One Identity同步账号数据。One Identity配置方法，请参见Setting up Synchronization with a Cloud Application。
- SCIM API：您可以通过SCIM API将企业内部系统中的账号映射成RAM用户，进行创建、删除、查询和修改操作。支持对RAM用户的如下字段进行操作：
  - `id`：RAM用户ID，全局唯一，服务端生成。

- `externalId`：RAM用户外键，用户级别唯一，客户端指定，用于将阿里云的RAM用户和企业内部系统中的用户关联。

  > **说明** 通过控制台创建的RAM用户无 `externalId` 字段。

- `userName`：RAM用户名称，用户级别唯一，客户端指定。
- `displayName`：RAM用户显示名称，客户端指定。

通过阿里云SCIM API同步账号数据的操作如下所示。

1. 获取已授权的ServerApp的ID（client_id）和密钥内容（client_secret）。
   - client_id：应用ID。从步骤一：创建OAuth应用并授权的步骤获取。
   - client_secret：应用密钥内容。从步骤一：创建OAuth应用并授权的步骤获取。
2. 通过 `client_id` 和 `client_secret` 访问 `https://oauth.aliyun.com/v1/token`，获取 `access_token`。

   其中，请求头中的Authorization内容为 `"Authorization: Basic Base64Encode(client_id:client_secret)"`，例如：cliet_id=cid，client_secret=123456，那么请求头中需要输入的Authorization内容为 `"Authorization: Basic Y2lkOjEyMzQ1Ng=="`。

   请求示例：

   ```
   curl --location --request POST --header "Authorization: Basic Y2lkOjEyMzQ1Ng==" https://oauth.aliyun.com/v1/token?grant_type=client_credentials&client_id=463790568674183****
   ```

   返回示例：

   ```
   {
       "scope": "/acs/scim",
       "request_id": "8dc768e0-d6fe-4f52-a788-05631dd6c584",
       "access_token": "eyJ***hKg",
       "token_type": "Bearer",
       "expires_in": "3599"
   }
   ```

3. 查询阿里云SCIM支持的 `ResourceType` 和 `Schema`。

   查询SCIM支持的 `ResourceType`：

   ```
   curl --location --request GET 'https://scim.aliyun.com/ResourceTypes'
   ```

   查询SCIM支持的 `Schema`：

   ```
   curl --location --request GET 'https://scim.aliyun.com/Schemas'
   ```

4. 创建、查询、修改和删除RAM用户。
   - 创建RAM用户。

     根据企业内部系统中的账号信息，指定与之对应的RAM用户名称（userName）、显示名称（displayName）和外键（externalId），然后创建RAM用户。

     请求示例：

```
curl --location --request POST 'https://scim.aliyun.com/Users' \
--header 'Authorization: Bearer  eyJ***hKg' \
--header 'Content-Type: application/json' \
--data-raw '{
    "displayName": "j2gg0s_****",
    "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:User"
    ],
    "externalId": "6e74eec4-ddb5-4e74-bd12-5e7b99b2****",
    "userName": "j2gg0screatedbyscim_exa****"
}'
```

返回示例：

```
{
    "displayName": "j2gg0s_****",
    "meta": {
        "created": "2020-02-14T03:58:59Z",
        "location": "https://scim.aliyun.com/Users/27648498165273****",
        "lastModified": "2020-02-14T03:58:59Z",
        "resourceType": "User"
    },
    "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:User"
    ],
    "externalId": "6e74eec4-ddb5-4e74-bd12-5e7b99b2****",
    "id": "27648498165273****",
    "userName": "j2gg0screatedbyscim_exa****"
}
```

- 查询指定的RAM用户。
  - 通过 `GET /Users/{id}` ，查询指定 `id` 的RAM用户。
  - 通过 `GET /Users?filter=externalId eq xxx` ，查询指定 `externalId` 的RAM用户。

- 通过 `GET /Users?filter=userName eq xxx` ，查询指定 `userName` 的RAM用户。请求示例和响应示例如下。

  请求示例：

```
curl --location --request GET 'https://scim.aliyun.com/Users?filter=userName%20eq%20%22j2gg0screatedbyscim****%22' \
--header 'Authorization: Bearer  eyJ***hKg'
```

  返回示例：

```
{
    "startIndex": 1,
    "totalResults": 1,
    "itemsPerPage": 30,
    "schemas": [
        "urn:ietf:params:scim:api:messages:2.0:ListResponse"
    ],
    "Resources": [
        {
            "displayName": "j2gg0screatedbyscim****",
            "meta": {
                "created": "2019-12-11T01:53:19Z",
                "location": "https://scim.aliyun.com/Users/27769827602919****",
                "lastModified": "2019-12-11T02:10:39Z",
                "resourceType": "User"
            },
            "schemas": [
                "urn:ietf:params:scim:schemas:core:2.0:User"
            ],
            "externalId": "6e74eec4-ddb5-4e74-bd12-5e7b99b2****",
            "id": "27769827602919****",
            "userName": "j2gg0screatedbyscim****"
        }
    ]
}
```

  说明 对于SCIM Filter，阿里云仅支持对 `id` 、 `userName` 和 `externalId` 进行and和eq操作。

- 修改RAM用户属性。

  请求示例：

```
curl --location --request PUT 'https://scim.aliyun.com/Users/27648498165273****' \
--header 'Authorization: Bearer  eyJ***hKg' \
--header 'Content-Type: application/json' \
--data-raw '{
    "displayName": "j2gg0s_new_****",
    "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:User"
    ],
    "externalId": "6e74eec4-ddb5-4e74-bd12-5e7b99b2****",
    "userName": "j2gg0screatedbyscim_new_exa****"
}'
```

返回示例：

```
{
    "displayName": "j2gg0s_new_****",
    "meta": {
        "created": "2020-02-14T03:58:59Z",
        "location": "https://scim.aliyun.com/Users/27648498165273****",
        "lastModified": "2020-02-14T04:03:55Z",
        "resourceType": "User"
    },
    "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:User"
    ],
    "externalId": "6e74eec4-ddb5-4e74-bd12-5e7b99b2****",
    "id": "27648498165273****",
    "userName": "j2gg0screatedbyscim_new_exa****"}
```

- 删除RAM用户。

  请求示例：

```
curl --location --request DELETE 'https://scim.aliyun.com/Users/27648498165273****' \
--header 'Authorization: Bearer  eyJ***hKg' \
--header 'Content-Type: application/json'
```

  返回的HTTP状态码为204时，表示删除RAM用户成功。

> **说明**
>
> - 如果RAM用户有关联的AccessKey、RAM Policy等，考虑到安全因素，不支持通过SCIM API直接删除RAM用户。
> - 阿里云暂时不支持软删除。如果您的系统支持软删除，建议先将软删除映射成硬删除，再同步到阿里云。

3.管理OAuth应用

3.1. 创建应用

您可以通过OAuth来创建Web应用、Native应用或Server应用，从而获取用户信息或访问阿里云API。

操作步骤

1. 使用阿里云账号登录RAM控制台。
2. 在左侧导航栏，单击**OAuth应用管理**。
3. 在**企业应用**页签，单击**创建应用**。
4. 在**创建应用**面板，设置应用参数。
   i. 输入**应用名称**和**显示名称**。
   ii. 选择**应用类型**。
      - **WebApp**： 指基于浏览器交互的网络应用。
      - **NativeApp**：指操作系统中运行的本地应用，主要为运行在桌面操作系统或移动操作系统中的应用。
      - **ServerApp**：指直接访问阿里云服务，而无需依赖用户登录的应用，目前仅支持基于SCIM协议的用户同步应用。示例请参见通过SCIM协议将企业内部账号同步到阿里云RAM。
   iii. 设置**访问令牌有效期**时长。

      访问令牌有效期范围：900秒（15分钟）~10,800秒（3小时）。默认值为3600秒。
   iv. 对于WebApp和NativeApp，设置**刷新令牌有效期**和**回调地址**。

      刷新令牌有效期范围：7200秒（2小时）~31,536,000秒（1年）。默认值为2,592,000秒。
5. 单击**保存**。

3.2. 查看应用基本信息

本文为您介绍如何查看应用基本信息，包括应用名称、显示名称和应用类型等信息。

操作步骤

1. 使用阿里云账号登录RAM控制台。
2. 在左侧导航栏，单击**OAuth应用管理**。
3. 在**企业应用**页签，单击目标应用名称。
4. 在**基本信息**区域，查看**应用名称**、**显示名称**和**应用类型**等信息。

3.3. 修改应用基本信息

本文为您介绍如何修改应用基本信息，包括应用的显示名称和访问令牌有效期等信息。

操作步骤

1. 使用阿里云账号登录RAM控制台。
2. 在左侧导航栏，单击**OAuth应用管理**。
3. 在**企业应用**页签，单击目标应用名称。

4. 在**基本信息**区域，单击**编辑基本信息**。
5. 修改**显示名称**和**访问令牌有效期**等信息，然后单击**保存**。

3.4. 添加应用范围

OAuth服务通过为应用添加范围来限定应用扮演用户登录阿里云后的权限。

操作步骤

1. 使用阿里云账号登录RAM控制台。
2. 在左侧导航栏，单击**OAuth应用管理**。
3. 在**企业应用**页签，单击目标应用名称。
4. 在**应用OAuth范围**页签，单击**添加OAuth范围**。
5. 在**添加OAuth范围**面板，选择需要添加的范围。

> **说明** *aliuid*和*profile*范围是与身份令牌相关，其他范围都与访问令牌相关。

6. 单击**确定**。

3.5. 创建应用密钥

如果应用需要访问阿里云API，需要创建应用密钥用作换取访问令牌时鉴定应用身份的密码。

操作步骤

1. 使用阿里云账号登录RAM控制台。
2. 在左侧导航栏，单击**OAuth应用管理**。
3. 在**企业应用**页签，单击目标应用名称。
4. 单击**应用密钥**页签，然后单击**创建密钥**。
5. 在**创建应用密钥**对话框，查看并复制创建成功的应用密钥，然后单击**关闭**。

> **说明**
> - 应用密钥内容仅在创建时可见，不支持查询，请及时保存。
> - 每个应用最多允许创建2个应用密钥。

3.6. 删除应用

如果不再需要使用应用获取用户信息或访问阿里云API，可以删除应用。删除应用后，企业用户将不能正常登录。

操作步骤

1. 使用阿里云账号登录RAM控制台。
2. 在左侧导航栏，单击**OAuth应用管理**。
3. 在**OAuth应用管理**页面，删除应用。
   - 删除自建的企业应用：先单击**企业应用**页签，然后单击目标应用**操作**列的**删除**，最后单击**确定**。

- 删除外部的第三方应用：先单击**第三方应用**页签，然后单击目标应用**操作**列的**删除**，最后单击**确定**。

4.OAuth常用的SDK示例

本文基于Spring Boot OAuth2和Pac4J，为您介绍OAuth常用的SDK示例的相关配置。

Spring Boot OAuth2示例

参考Spring Boot and OAuth2文档及示例，主要做以下两点修改：

- 配置文件改为阿里云对应的配置。

```
aliyun:
    client:
      clientId: 415195082384692****
      clientSecret: 6EwN4qutnZuchG6n677Ie33SsjAhwyTpcOMSoIo6v0gqJtw4QcHhERVXfqzc****
      accessTokenUri: https://oauth.aliyun.com/v1/token
      userAuthorizationUri: https://signin.aliyun.com/oauth2/v1/auth
      tokenName: access_token
      authenticationScheme: query
      clientAuthenticationScheme: form
    resource:
      userInfoUri: https://oauth.aliyun.com/v1/userinfo
```

- 修改重定向URI。

  修改 `OAuth2ClientAuthenticationProcessingFilter` 中的回调地址，改成与应用中的配置相同。 例如：应用配置的是 `http://localhost:8080/login/aliyun` ，则把代码中的回调地址替换为 `/login/aliyun` 。

  整体示例代码如下：

```
public class WebApplication extends WebSecurityConfigurerAdapter {
  @Autowired
  OAuth2ClientContext oauth2ClientContext;
  @RequestMapping("/user")
  public Principal user(Principal principal) {
    return principal;
  }
  @Override
  protected void configure(HttpSecurity http) throws Exception {
    // @formatter:off
    http.antMatcher("/**").authorizeRequests().antMatchers("/", "/login**", "/webjars/**"
).permitAll().anyRequest()
        .authenticated().and().exceptionHandling()
        .authenticationEntryPoint(new LoginUrlAuthenticationEntryPoint("/")).and().logout
()
        .logoutSuccessUrl("/").permitAll().and().csrf()
        .csrfTokenRepository(CookieCsrfTokenRepository.withHttpOnlyFalse()).and()
        .addFilterBefore(ssoFilter(), BasicAuthenticationFilter.class);
    // @formatter:on
  }
  public static void main(String[] args) {
    SpringApplication.run(WebApplication.class, args);
  }
  @Bean
  public FilterRegistrationBean oauth2ClientFilterRegistration(OAuth2ClientContextFilter
```

```
filter) {
    FilterRegistrationBean registration = new FilterRegistrationBean();
    registration.setFilter(filter);
    registration.setOrder(-100);
    return registration;
  }
  private Filter ssoFilter() {
    OAuth2ClientAuthenticationProcessingFilter aliyunFilter = new OAuth2ClientAuthenticat
ionProcessingFilter(
        "/login/aliyun");
    OAuth2RestTemplate aliyunTemplate = new OAuth2RestTemplate(aliyun(), oauth2ClientCont
ext);
    aliyunFilter.setRestTemplate(aliyunTemplate);
    UserInfoTokenServices tokenServices = new UserInfoTokenServices(aliyunResource().getU
serInfoUri(),
        aliyun().getClientId());
    tokenServices.setRestTemplate(aliyunTemplate);
    aliyunFilter.setTokenServices(tokenServices);
    return aliyunFilter;
  }
  @Bean
  @ConfigurationProperties("aliyun.client")
  public AuthorizationCodeResourceDetails aliyun() {
    return new AuthorizationCodeResourceDetails();
  }
  @Bean
  @ConfigurationProperties("aliyun.resource")
  public ResourceServerProperties aliyunResource() {
    return new ResourceServerProperties();
  }
}
```

Pac4J 示例

参考Pac4J的spring-webmvc-pac4j示例，进行以下操作：

- 创建 `AliyunOidcClient` 。

```
public class AliyunOidcClient extends OidcClient<OidcProfile, OidcConfiguration> {
    public AliyunOidcClient() {
    }
    public AliyunOidcClient(OidcConfiguration configuration) {
        super(configuration);
    }
    @Override
    protected void clientInit() {
        CommonHelper.assertNotNull("configuration", this.getConfiguration());
        this.getConfiguration().defaultDiscoveryURI("https://oauth.aliyun.com/.well-known
/openid-configuration");
        OidcProfileCreator<OidcProfile> profileCreator = new OidcProfileCreator(this.getC
onfiguration());
        profileCreator.setProfileDefinition(new OidcProfileDefinition((x) -> {
            return new OidcProfile();
        }));
        this.defaultProfileCreator(profileCreator);
        super.clientInit();
    }
}
```

- 添加 `oidcConfig` 的*bean*。

```
<bean id="oidcConfiguration" class="org.pac4j.oidc.config.OidcConfiguration">
        <property name="clientId" value=your application id />
        <property name="secret" value=your application secret />
        <property name="useNonce" value="false" />
        <property name="scope" value="openid profile aliuid" />
        <property name="clientAuthenticationMethod" value="client_secret_post" />
</bean>
```

- 添加 `aliyunOidcClient` 的*bean*。

```
<bean id="aliyunOidClient" class="org.pac4j.demo.spring.AliyunOidcClient">
        <constructor-arg name="configuration" ref="oidcConfiguration" />
        <property name="authorizationGenerator">
            <bean class="org.pac4j.demo.spring.RoleAdminAuthGenerator" />
        </property>
</bean>
```

- 配置 `bean clients` 的*callbackUrl*以及*client*属性。

> **说明** 其中*callbackUrl*需要配置在阿里云控制台中应用的回调地址里。

```
<bean id="clients" class="org.pac4j.core.client.Clients">
        <constructor-arg name="callbackUrl" value="http://127.0.0.1:8080/callback" />
        <constructor-arg name="clients">
            <list>
                <ref bean="aliyunOidClient" />
            </list>
        </constructor-arg>
</bean>
```

- 添加 `oidc` 拦截器。

```
<mvc:interceptor>
            <mvc:mapping path="/oidc/*" />
            <bean class="org.pac4j.springframework.web.SecurityInterceptor">
                <constructor-arg name="config" ref="config" />
                <constructor-arg name="clients" value="AliyunOidcClient" />
            </bean>
</mvc:interceptor>
```

- 启动示例，访问 `http://localhost:8080/oidc/index.html` 。