阿里云
访问控制

开放授权管理（OAuth）

文档版本：20190920

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
	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。	 **禁止:** 重置操作将丢失用户配置数据。
	该类警示信息可能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，恢复业务所需时间约10分钟。
	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。	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。	单击 确定。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
*## *	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。	`swich` *`{stand | slave}`*

目录

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

为了更好的理解OAuth 2.0协议，下面简要介绍与OAuth 2.0相关的一些基本概念：

用户	此处用户可以是主账号也可以是RAM用户，用户需要登录到阿里云并授权应用访问阿里云资源。
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。
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。 目前OAuth支持的范围如下： · `openid`：获取登录用户的基本信息（默认授权域，不可移除）。 说明： OpenID Connect（OIDC）协议的默认范围，所有的应用默认具有这一范围，不需要额外添加。 · `aliuid`：阿里云颁发的唯一用户标志符。 · `profile`：用户名称等个人信息。 · `/acs/ccc`：阿里云呼叫中心服务API。 · `/acs/alidns`：阿里云解析API。 说明： `openid`、`aliuid`以及`profile`这几个范围与身份令牌相关，其他范围都与访问令牌相关。

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：身份令牌只包含用户的身份信息，不能用于访问阿里云资源。 · 访问令牌：访问令牌包含了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：刷新令牌可以用于换取新的访问令牌。
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

- #unique_4
- #unique_5
- #unique_6

2 OAuth应用典型场景

2.1 Web应用登录阿里云

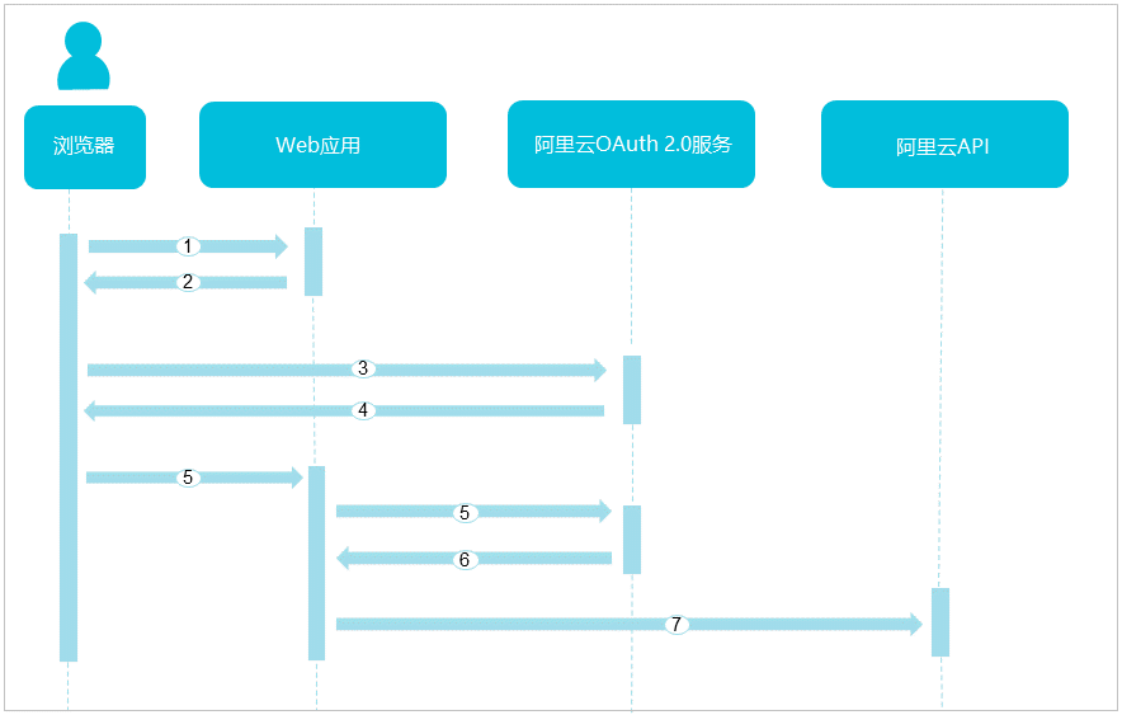
本文介绍Web应用如何通过OAuth 2.0扮演登录用户访问阿里云API。

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，并为应用生成应用密钥。详情请参见#unique_9。

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

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。
5. 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应用返回令牌。
7. Web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。

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`。

   表 2-1: 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。
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。

参数名称	是否必选	描述
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。

请求示例

```
https://signin.aliyun.com/oauth2/v1/auth?
client_id=123****
redirect_uri=https%3A%2F%2Fyourwebapp.com%2Fauthcallback%2F&
response_type=code&
scope=openid%20%2Facs%2Fccc&
access_type=offline&
state=123456****
```

返回示例

```
GET HTTP/1.1 302 Found
Location: https://yourwebapp.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`。

   表 2-2: 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。
client_secret	否	应用的密钥，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://yourwebapp.com/authcallback/&
grant_type=authorization_code
```

表 2-3: 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。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。

返回示例

```
{
  "access_token": "eyJraWQiOiJrMTIzNCIsImVu****",
  "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`。

表 2-4: 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。

参数名称	是否必选	描述
client_secret	否	应用的密钥，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
```

表 2-5: 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,
}
```

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

如果应用获取了刷新令牌，在特定的场景下也需要撤销刷新令牌，例如：用户退出登录应用或用户将自己的账号从应用中移除等。

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

表 2-6: 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。
client_secret	否	应用的密钥。

2.2 Native应用登录阿里云

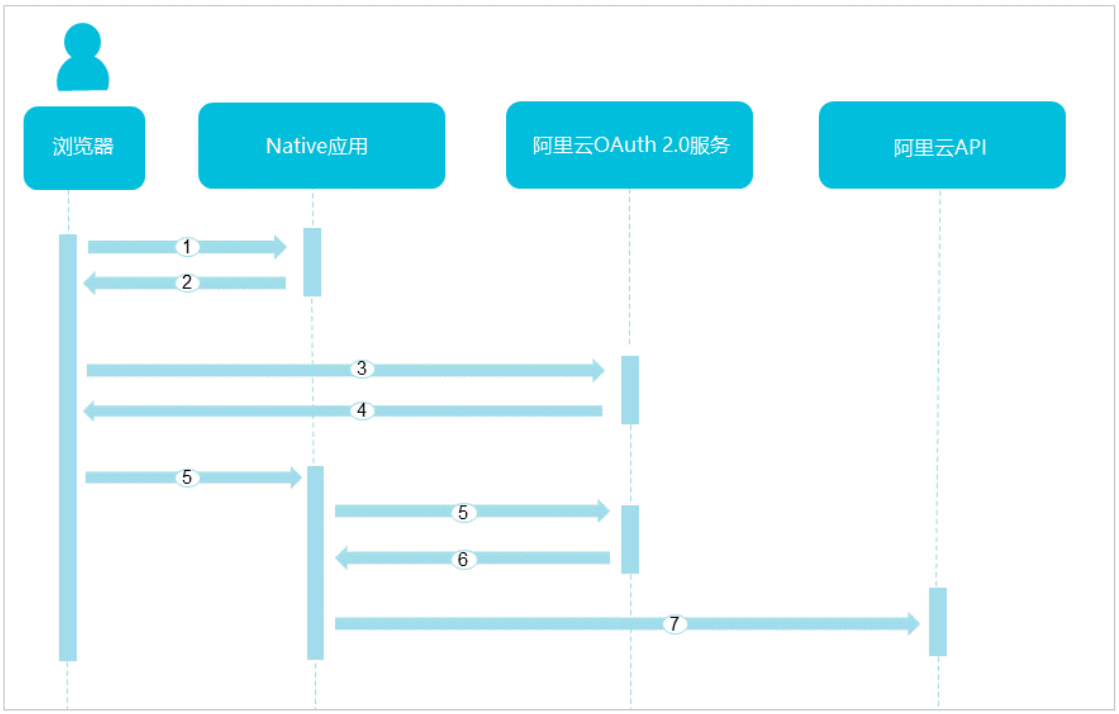
本文介绍桌面和移动端的Native应用如何通过OAuth 2.0扮演登录用户访问阿里云API。

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#unique_9。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。
5. 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。

方式	取值
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。

   表 2-7: 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。

参数名称	是否必选	描述
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`。

   表 2-8: 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
```

   表 2-9: 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。

参数名称	描述
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`。

表 2-10: 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
```

表 2-11: 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。

表 2-12: 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

应用获取用户登录信息首先需要创建应用，为应用提供恰当的名称、OAuth范围、回调地址等关键信息，并为应用生成应用密钥。详情请参见#unique_9。

OIDC基本概念

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。
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。

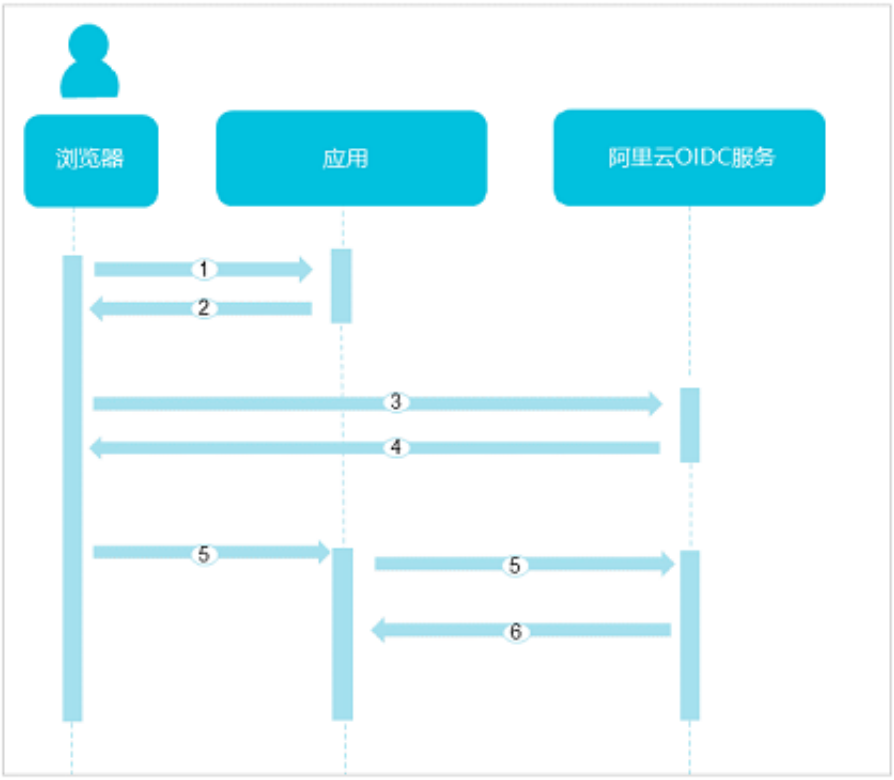
阿里云作为OIDC服务提供者，提供了一个discovery endpoint：`https://oauth.aliyun.com/.well-known/openid-configuration`来简化配置流程。

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。
5. 应用使用授权码向阿里云OIDC服务申请身份令牌。

   如何获取身份令牌的签名密钥，请参见应用获取身份令牌签名密钥。
6. 阿里云OIDC服务向应用返回身份令牌，应用通过身份令牌便可以获取用户信息。
   - 在身份令牌用于获取用户信息的场景下，由于身份令牌是由应用直接从OIDC服务获取的，因此应用不需要验证令牌的签名。

     用户信息返回示例请参见用户信息返回示例。
   - 在身份令牌用于应用的各个不同模块之间通信的场景下，为保证信息安全，建议应用接受对身份令牌进行验证。

     如何验证身份令牌，请参见验证身份令牌的JWT（JSON Web Token）签名。

应用获取身份令牌签名密钥

请求示例

```
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());
}
```

验证身份令牌的JWT（JSON Web Token）签名

阿里云颁发的身份令牌是带有签名的JWT，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。
- 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。

请求示例

```
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.toRSAPubli
cKey());
      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");
```

```
}
```

用户信息返回示例

表 2-13: 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

表 2-14: Body 返回参数

参数名称	描述	需要的 OAuth 范围
exp	令牌过期时间。	openid
sub	令牌主体，登录用户的 UID。	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。	openid
name	登录用户的显示名称。	profile
upn	登录用户的UPN（User Principal Name）。	profile
login_name	云账号的登录名。	profile
aid	登录用户所属的阿里云账号ID。	aliuid
uid	登录用户的阿里云账号ID。	aliuid

返回示例

返回 Header

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

返回Body

为了阅读方便，这里展示的是未加密的身份令牌的返回。实际返回为加密的身份令牌，详情请参见验证身份令牌的JWT（JSON Web Token）签名。

- 云账号请求时的返回Body

```
{
  "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", // 云账号的登录名
  "aid": "123456789012****", //对应的阿里云账号ID
  "uid": "123456789012****", //登录用户的阿里云账号ID
}
```

- RAM 用户请求时的返回 Body

```
{
  "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.com",// RAM用户的登录名
  "aid": "123456789012****", //对应的阿里云账号ID
  "uid": "234567890123****", //登录用户的阿里云账号ID
}
```

更多信息

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，返回信息不加密。

说明:

OIDC场景下，即只有 openidaliuid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。

获取用户信息的请求地址：`https://oauth.aliyun.com/v1/userinfo`。

请求示例

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

返回 Body

- 云账号请求时的返回 Body

```
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", // 云账号的登录名
  "aid": "123456789012****", //对应的阿里云账号ID
  "uid": "123456789012****", //登录用户的阿里云账号ID
}
```

- **RAM用户请求时的返回Body**

```
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.com",// RAM用户的登录名
  "aid": "123456789012****", //对应的阿里云账号ID
  "uid": "234567890123****", //登录用户的阿里云账号ID
}
```

3 管理OAuth应用

3.1 创建应用

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

操作步骤

1. 云账号登录RAM控制台。
2. 在左侧导航栏，单击OAuth应用管理。
3. 单击新建应用。
4. 输入应用名称和应用显示名称。
5. 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。
6. 根据需要修改访问令牌有效期时长。

   说明：
   有效期可设置范围为：15分钟至3小时，默认为3600秒。
7. 根据需要修改刷新令牌有效期时长。

   说明：
   有效期可设置范围为：2小时至1年，默认为2592000秒。
8. 输入回调地址。
9. 单击确认。

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. 云账号登录RAM控制台。
2. 在左侧导航栏，单击OAuth应用管理。
3. 在应用名称列表下，单击目标应用名称。

4. 在基本信息区域下，可以查看应用基本信息。

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，包括应用名称、应用显示名称和令牌有效期等信息。

操作步骤

1. 云账号登录RAM控制台。
2. 在左侧导航栏，单击OAuth应用管理。
3. 在应用名称列表下，单击目标应用名称。
4. 在基本信息区域下，单击编辑基本信息。
5. 修改完成后，单击确认。

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. 云账号登录RAM控制台。
2. 在左侧导航栏，单击OAuth应用管理。
3. 在应用名称列表下，单击目标应用名称。
4. 在应用OAuth范围页签下，单击添加OAuth范围。
5. 从左侧范围列表下，勾选需要的范围。

   说明:

   `openid`、`aliuid`和`profile`这几个范围与身份令牌相关，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. 云账号登录RAM控制台。
2. 在左侧导航栏，单击OAuth应用管理。
3. 在应用名称列表下，单击目标应用名称。
4. 在应用密钥页签下，单击创建密钥。

5. 在弹出的对话框中查看应用密钥，单击确认。

> **说明:**
> 最多可以创建2个应用密钥。应用密钥只在创建时显示，不支持查询，请妥善保管。

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. 云账号登录RAM控制台。
2. 在左侧导航栏，单击OAuth应用管理。
3. 在应用名称列表下，找到目标应用，单击删除。
4. 单击确认。

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: 6EwN4qutnZuchG6n677Ie33SsjAhwyTpcOMSoIo6
v0gqJtw4QcHhERVXfqzc****
      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 LoginUrlAuthenticationEntryPoi
nt("/")).and().logout()
        .logoutSuccessUrl("/").permitAll().and().csrf()
        .csrfTokenRepository(CookieCsrfTokenRepository.withHttpOn
lyFalse()).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
OAuth2ClientAuthenticationProcessingFilter(
        "/login/aliyun");
    OAuth2RestTemplate aliyunTemplate = new OAuth2RestTemplate(
aliyun(), oauth2ClientContext);
    aliyunFilter.setRestTemplate(aliyunTemplate);
    UserInfoTokenServices tokenServices = new UserInfoTokenServices(
aliyunResource().getUserInfoUri(),
        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.getConfigu
ration());
        this.getConfiguration().defaultDiscoveryURI("https://oauth.
aliyun.com/.well-known/openid-configuration");
        OidcProfileCreator<OidcProfile> profileCreator = new
OidcProfileCreator(this.getConfiguration());
        profileCreator.setProfileDefinition(new OidcProfileDefinitio
n((x) -> {
            return new OidcProfile();
        }));
        this.defaultProfileCreator(profileCreator);
        super.clientInit();
    }
}
```

- 添加oidcConfig的bean。

```
<bean id="oidcConfiguration" class="org.pac4j.oidc.config.OidcConfig
uration">
        <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.AliyunOidc
Client">
        <constructor-arg name="configuration" ref="oidcConfiguration
" />
        <property name="authorizationGenerator">
            <bean class="org.pac4j.demo.spring.RoleAdminAuthGenerat
or" />
        </property>
</bean>
```

- 配置bean clients的callbackUrl以及client属性。

说明:

其中callbackUri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.SecurityIn
terceptor">
                <constructor-arg name="config" ref="config" />
                <constructor-arg name="clients" value="AliyunOidc
Client" />
            </bean>
</mvc:interceptor>
```

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