<http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html>

# 理解OAuth 2.0

作者： [阮一峰](http://www.ruanyifeng.com/)

日期： [2014年5月12日](http://www.ruanyifeng.com/blog/2014/05/)

[](http://www.zhufengpeixun.cn/main/index.html?ref=ruanyifeng)

[OAuth](http://en.wikipedia.org/wiki/OAuth)是一个关于授权（authorization）的开放网络标准，在全世界得到广泛应用，目前的版本是2.0版。

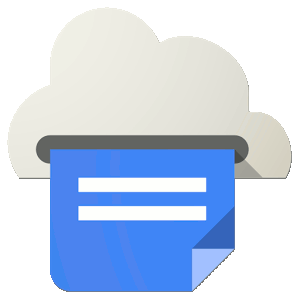
本文对OAuth 2.0的设计思路和运行流程，做一个简明通俗的解释，主要参考材料为[RFC 6749](http://www.rfcreader.com/#rfc6749)。



## 一、应用场景

为了理解OAuth的适用场合，让我举一个假设的例子。

有一个"云冲印"的网站，可以将用户储存在Google的照片，冲印出来。用户为了使用该服务，必须让"云冲印"读取自己储存在Google上的照片。



问题是只有得到用户的授权，Google才会同意"云冲印"读取这些照片。那么，"云冲印"怎样获得用户的授权呢？

传统方法是，用户将自己的Google用户名和密码，告诉"云冲印"，后者就可以读取用户的照片了。这样的做法有以下几个严重的缺点。

（1）"云冲印"为了后续的服务，会保存用户的密码，这样很不安全。

（2）Google不得不部署密码登录，而我们知道，单纯的密码登录并不安全。

（3）"云冲印"拥有了获取用户储存在Google所有资料的权力，用户没法限制"云冲印"获得授权的范围和有效期。

（4）用户只有修改密码，才能收回赋予"云冲印"的权力。但是这样做，会使得其他所有获得用户授权的第三方应用程序全部失效。

（5）只要有一个第三方应用程序被破解，就会导致用户密码泄漏，以及所有被密码保护的数据泄漏。

OAuth就是为了解决上面这些问题而诞生的。

## 二、名词定义

在详细讲解OAuth 2.0之前，需要了解几个专用名词。它们对读懂后面的讲解，尤其是几张图，至关重要。

（1） **Third-party application**：第三方应用程序，本文中又称"客户端"（client），即上一节例子中的"云冲印"。

（2）**HTTP service**：HTTP服务提供商，本文中简称"服务提供商"，即上一节例子中的Google。

（3）**Resource Owner**：资源所有者，本文中又称"用户"（user）。

（4）**User Agent**：用户代理，本文中就是指浏览器。

（5）**Authorization server**：认证服务器，即服务提供商专门用来处理认证的服务器。

（6）**Resource server**：资源服务器，即服务提供商存放用户生成的资源的服务器。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。

知道了上面这些名词，就不难理解，OAuth的作用就是让"客户端"安全可控地获取"用户"的授权，与"服务商提供商"进行互动。

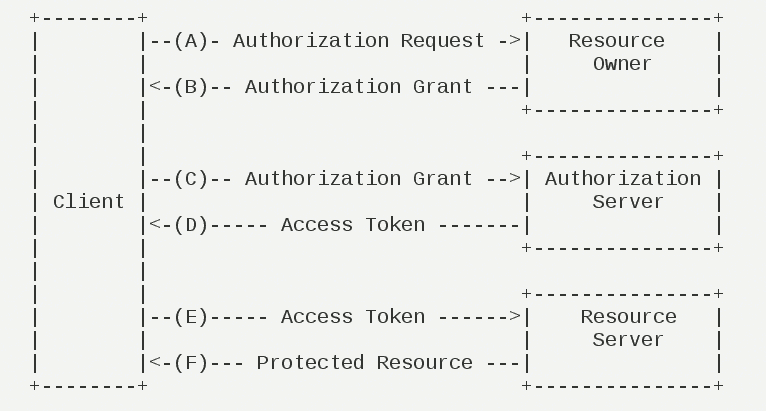
## 三、OAuth的思路

OAuth在"客户端"与"服务提供商"之间，设置了一个授权层（authorization layer）。"客户端"不能直接登录"服务提供商"，只能登录授权层，以此将用户与客户端区分开来。"客户端"登录授权层所用的令牌（token），与用户的密码不同。用户可以在登录的时候，指定授权层令牌的权限范围和有效期。

"客户端"登录授权层以后，"服务提供商"根据令牌的权限范围和有效期，向"客户端"开放用户储存的资料。

## 四、运行流程

OAuth 2.0的运行流程如下图，摘自RFC 6749。



（A）用户打开客户端以后，客户端要求用户给予授权。

（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，同意发放令牌。

（E）客户端使用令牌，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。

不难看出来，上面六个步骤之中，B是关键，即用户怎样才能给于客户端授权。有了这个授权以后，客户端就可以获取令牌，进而凭令牌获取资源。

下面一一讲解客户端获取授权的四种模式。

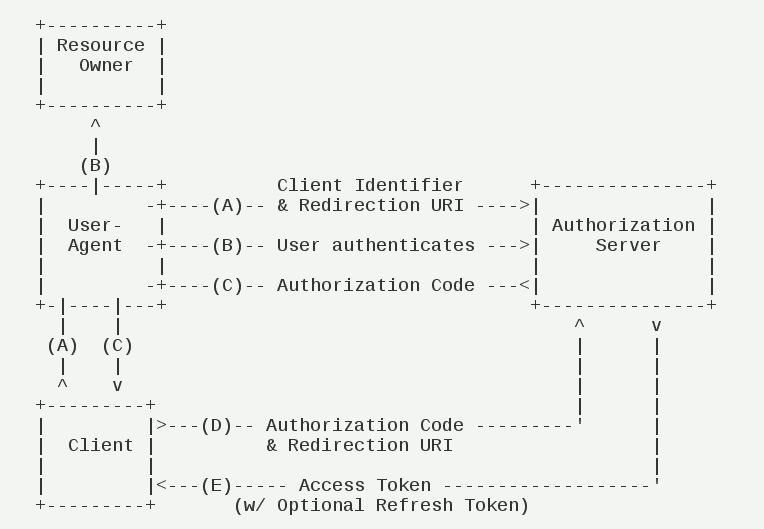
## 五、客户端的授权模式

客户端必须得到用户的授权（authorization grant），才能获得令牌（access token）。OAuth 2.0定义了四种授权方式。

* 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）

## 六、授权码模式

授权码模式（authorization code）是功能最完整、流程最严密的授权模式。它的特点就是通过客户端的后台服务器，与"服务提供商"的认证服务器进行互动。



它的步骤如下：

（A）用户访问客户端，后者将前者导向认证服务器。

（B）用户选择是否给予客户端授权。

（C）假设用户给予授权，认证服务器将用户导向客户端事先指定的"重定向URI"（redirection URI），同时附上一个授权码。

（D）客户端收到授权码，附上早先的"重定向URI"，向认证服务器申请令牌。这一步是在客户端的后台的服务器上完成的，对用户不可见。

（E）认证服务器核对了授权码和重定向URI，确认无误后，向客户端发送访问令牌（access token）和更新令牌（refresh token）。

下面是上面这些步骤所需要的参数。

A步骤中，客户端申请认证的URI，包含以下参数：

* response\_type：表示授权类型，必选项，此处的值固定为"code"
* client\_id：表示客户端的ID，必选项
* redirect\_uri：表示重定向URI，可选项
* scope：表示申请的权限范围，可选项
* state：表示客户端的当前状态，可以指定任意值，认证服务器会原封不动地返回这个值。

下面是一个例子。

GET /authorize?response\_type=code&client\_id=s6BhdRkqt3&state=xyz

&redirect\_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1

Host: server.example.com

C步骤中，服务器回应客户端的URI，包含以下参数：

* code：表示授权码，必选项。该码的有效期应该很短，通常设为10分钟，客户端只能使用该码一次，否则会被授权服务器拒绝。该码与客户端ID和重定向URI，是一一对应关系。
* state：如果客户端的请求中包含这个参数，认证服务器的回应也必须一模一样包含这个参数。

下面是一个例子。

HTTP/1.1 302 Found

Location: <https://client.example.com/cb>?code=SplxlOBeZQQYbYS6WxSbIA

&state=xyz

D步骤中，客户端向认证服务器申请令牌的HTTP请求，包含以下参数：

* grant\_type：表示使用的授权模式，必选项，此处的值固定为"authorization\_code"。
* code：表示上一步获得的授权码，必选项。
* redirect\_uri：表示重定向URI，必选项，且必须与A步骤中的该参数值保持一致。
* client\_id：表示客户端ID，必选项。

下面是一个例子。

POST /token HTTP/1.1

Host: server.example.com

Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

Content-Type: application/x-www-form-urlencoded

grant\_type=authorization\_code&code=SplxlOBeZQQYbYS6WxSbIA

&redirect\_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb

E步骤中，认证服务器发送的HTTP回复，包含以下参数：

* access\_token：表示访问令牌，必选项。
* token\_type：表示令牌类型，该值大小写不敏感，必选项，可以是bearer类型或mac类型。
* expires\_in：表示过期时间，单位为秒。如果省略该参数，必须其他方式设置过期时间。
* refresh\_token：表示更新令牌，用来获取下一次的访问令牌，可选项。
* scope：表示权限范围，如果与客户端申请的范围一致，此项可省略。

下面是一个例子。

HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8

Cache-Control: no-store

Pragma: no-cache

{

"access\_token":"2YotnFZFEjr1zCsicMWpAA",

"token\_type":"example",

"expires\_in":3600,

"refresh\_token":"tGzv3JOkF0XG5Qx2TlKWIA",

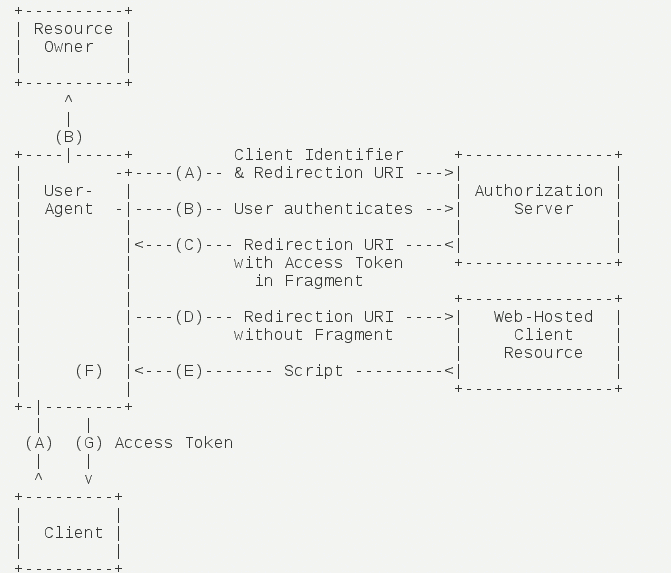
"example\_parameter":"example\_value"

}

从上面代码可以看到，相关参数使用JSON格式发送（Content-Type: application/json）。此外，HTTP头信息中明确指定不得缓存。

## 七、简化模式

简化模式（implicit grant type）不通过第三方应用程序的服务器，直接在浏览器中向认证服务器申请令牌，跳过了"授权码"这个步骤，因此得名。所有步骤在浏览器中完成，令牌对访问者是可见的，且客户端不需要认证。



它的步骤如下：

（A）客户端将用户导向认证服务器。

（B）用户决定是否给于客户端授权。

（C）假设用户给予授权，认证服务器将用户导向客户端指定的"重定向URI"，并在URI的Hash部分包含了访问令牌。

（D）浏览器向资源服务器发出请求，其中不包括上一步收到的Hash值。

（E）资源服务器返回一个网页，其中包含的代码可以获取Hash值中的令牌。

（F）浏览器执行上一步获得的脚本，提取出令牌。

（G）浏览器将令牌发给客户端。

下面是上面这些步骤所需要的参数。

A步骤中，客户端发出的HTTP请求，包含以下参数：

* response\_type：表示授权类型，此处的值固定为"token"，必选项。
* client\_id：表示客户端的ID，必选项。
* redirect\_uri：表示重定向的URI，可选项。
* scope：表示权限范围，可选项。
* state：表示客户端的当前状态，可以指定任意值，认证服务器会原封不动地返回这个值。

下面是一个例子。

GET /authorize?response\_type=token&client\_id=s6BhdRkqt3&state=xyz

&redirect\_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1

Host: server.example.com

C步骤中，认证服务器回应客户端的URI，包含以下参数：

* access\_token：表示访问令牌，必选项。
* token\_type：表示令牌类型，该值大小写不敏感，必选项。
* expires\_in：表示过期时间，单位为秒。如果省略该参数，必须其他方式设置过期时间。
* scope：表示权限范围，如果与客户端申请的范围一致，此项可省略。
* state：如果客户端的请求中包含这个参数，认证服务器的回应也必须一模一样包含这个参数。

下面是一个例子。

HTTP/1.1 302 Found

Location: <http://example.com/cb>#access\_token=2YotnFZFEjr1zCsicMWpAA

&state=xyz&token\_type=example&expires\_in=3600

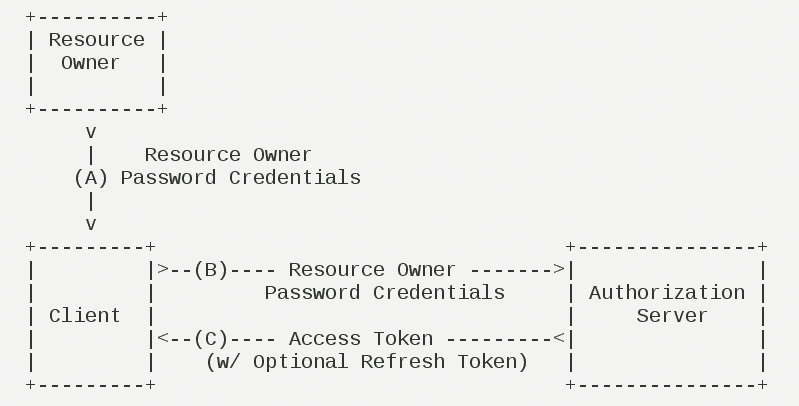
在上面的例子中，认证服务器用HTTP头信息的Location栏，指定浏览器重定向的网址。注意，在这个网址的Hash部分包含了令牌。

根据上面的D步骤，下一步浏览器会访问Location指定的网址，但是Hash部分不会发送。接下来的E步骤，服务提供商的资源服务器发送过来的代码，会提取出Hash中的令牌。

## 八、密码模式

密码模式（Resource Owner Password Credentials Grant）中，用户向客户端提供自己的用户名和密码。客户端使用这些信息，向"服务商提供商"索要授权。

在这种模式中，用户必须把自己的密码给客户端，但是客户端不得储存密码。这通常用在用户对客户端高度信任的情况下，比如客户端是操作系统的一部分，或者由一个著名公司出品。而认证服务器只有在其他授权模式无法执行的情况下，才能考虑使用这种模式。



它的步骤如下：

（A）用户向客户端提供用户名和密码。

（B）客户端将用户名和密码发给认证服务器，向后者请求令牌。

（C）认证服务器确认无误后，向客户端提供访问令牌。

B步骤中，客户端发出的HTTP请求，包含以下参数：

* grant\_type：表示授权类型，此处的值固定为"password"，必选项。
* username：表示用户名，必选项。
* password：表示用户的密码，必选项。
* scope：表示权限范围，可选项。

下面是一个例子。

POST /token HTTP/1.1

Host: server.example.com

Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

Content-Type: application/x-www-form-urlencoded

grant\_type=password&username=johndoe&password=A3ddj3w

C步骤中，认证服务器向客户端发送访问令牌，下面是一个例子。

HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8

Cache-Control: no-store

Pragma: no-cache

{

"access\_token":"2YotnFZFEjr1zCsicMWpAA",

"token\_type":"example",

"expires\_in":3600,

"refresh\_token":"tGzv3JOkF0XG5Qx2TlKWIA",

"example\_parameter":"example\_value"

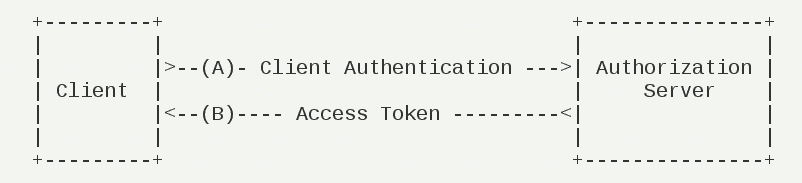
}

上面代码中，各个参数的含义参见《授权码模式》一节。

整个过程中，客户端不得保存用户的密码。

## 九、客户端模式

客户端模式（Client Credentials Grant）指客户端以自己的名义，而不是以用户的名义，向"服务提供商"进行认证。严格地说，客户端模式并不属于OAuth框架所要解决的问题。在这种模式中，用户直接向客户端注册，客户端以自己的名义要求"服务提供商"提供服务，其实不存在授权问题。



它的步骤如下：

（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。

（B）认证服务器确认无误后，向客户端提供访问令牌。

A步骤中，客户端发出的HTTP请求，包含以下参数：

* granttype：表示授权类型，此处的值固定为"clientcredentials"，必选项。
* scope：表示权限范围，可选项。

POST /token HTTP/1.1

Host: server.example.com

Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

Content-Type: application/x-www-form-urlencoded

grant\_type=client\_credentials

认证服务器必须以某种方式，验证客户端身份。

B步骤中，认证服务器向客户端发送访问令牌，下面是一个例子。

HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8

Cache-Control: no-store

Pragma: no-cache

{

"access\_token":"2YotnFZFEjr1zCsicMWpAA",

"token\_type":"example",

"expires\_in":3600,

"example\_parameter":"example\_value"

}

上面代码中，各个参数的含义参见《授权码模式》一节。

## 十、更新令牌

如果用户访问的时候，客户端的"访问令牌"已经过期，则需要使用"更新令牌"申请一个新的访问令牌。

客户端发出更新令牌的HTTP请求，包含以下参数：

* granttype：表示使用的授权模式，此处的值固定为"refreshtoken"，必选项。
* refresh\_token：表示早前收到的更新令牌，必选项。
* scope：表示申请的授权范围，不可以超出上一次申请的范围，如果省略该参数，则表示与上一次一致。

下面是一个例子。

POST /token HTTP/1.1

Host: server.example.com

Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

Content-Type: application/x-www-form-urlencoded

grant\_type=refresh\_token&refresh\_token=tGzv3JOkF0XG5Qx2TlKWIA

（完）

### 文档信息

* 版权声明：自由转载-非商用-非衍生-保持署名（[创意共享3.0许可证](http://creativecommons.org/licenses/by-nc-nd/3.0/deed.zh)）
* 发表日期： 2014年5月12日
* 更多内容： [档案](http://www.ruanyifeng.com/blog/archives.html) » [开发者手册](http://www.ruanyifeng.com/blog/developer/)
* 文集：[《前方的路》](http://www.ruanyifeng.com/road)，[《未来世界的幸存者》](http://www.ruanyifeng.com/survivor)
* 社交媒体：[twitter](https://twitter.com/ruanyf)，[weibo](http://weibo.com/ruanyf)

## 相关文章

* **2018.11.07:**[awk 入门教程](http://www.ruanyifeng.com/blog/2018/11/awk.html)

awk是处理文本文件的一个应用程序，几乎所有 Linux 系统都自带这个程序。

* **2018.11.05:**[Page Lifecycle API 教程](http://www.ruanyifeng.com/blog/2018/11/page_lifecycle_api.html)

两周前，我介绍了 Page Visibility API。有了它，就可以监听各种情况的网页卸载。

* **2018.10.25:**[Page Visibility API 教程](http://www.ruanyifeng.com/blog/2018/10/page_visibility_api.html)

一、简介 有时候，开发者需要知道，用户正在离开页面。常用的方法是监听下面三个事件。

* **2018.10.18:**[Flexbox 布局的最简单表单](http://www.ruanyifeng.com/blog/2018/10/flexbox-form.html)

弹性布局（Flexbox）逐渐流行，越来越多人使用，因为它写 CSS 布局真是太方便了。

## 广告[（购买广告位）](http://www.ruanyifeng.com/support.html)

[](https://gekiux.taobao.com/p/rd913665.htm)

[](http://www.ruanyifeng.com/support.html)

## 留言（154条）

[**周梦康**](http://mengkang.net/)**说：**

阮老师的文章都是精品，真正做到深入浅出。

2014年5月12日 22:10 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-322940) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**hutusi**](http://hutusi.com/)**说：**

留名，慢慢看~

2014年5月12日 22:58 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-322952) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**chility 说：**

今天培训老师讲了这部分，恰好阮老师写了这篇文章介绍，加深理解。

2014年5月13日 00:31 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-322957) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**phi 说：**

那什么，没看懂。能不能用位图。ASCII图看的很不习惯。谢谢

2014年5月13日 01:24 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-322960) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**WhatDoesTheFoxSay 说：**

授权码模式这部分中的“E步骤中，客户端发送的HTTP回复，包含以下参数“这句话，是否应为”认证服务器发送的HTTP回复“？

2014年5月13日 05:00 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-322994) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**Charles**](http://sexywp.com/)**说：**

本文关于state参数的解释，是语焉不详的。事实上，绝大多数的互联网中文文档对这个参数都语焉不详。我想，对于造成近期互联网上危害广泛的OAuth漏洞来说，众多中文技术资料对这个参数解释不到位，对这个参数的实现没有给出清晰的指导，也是成因的一部分。原文是这么说的：RECOMMENDED. An opaque value used by the client to maintain state between the request and callback. The authorization server includes this value when redirecting the user-agent back to the client. The parameter SHOULD be used for preventing cross-site request forgery as described in Section 10.12.如果阮老师只是翻译为什么“客户端状态”之类的话，还有什么“可以随便填”，这算不上一种到位的解释。首先，这个参数是“RECOMMENDED”，并非什么可有可无的东西，事实是很多厂商都实现成了可有可无，其次，这个参数是“SHOULD”，为啥“SHOULD”呢，因为会引发“CSRF”。

2014年5月13日 07:41 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-323002) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**阮一峰 说：**

**引用WhatDoesTheFoxSay的发言：**

授权码模式这部分中的“E步骤中，客户端发送的HTTP回复，包含以下参数“这句话，是否应为”认证服务器发送的HTTP回复“？

谢谢指出，已经改过来了。

2014年5月13日 08:44 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-323009) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**Tonni**](http://houcoder.github.io/)**说：**

阮兄的博客又更新了，前几天买的《黑客与画家》昨天到了，很不错的书。

2014年5月13日 09:05 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-323011) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**YG**](http://gangyou.me/)**说：**

看了这么多OAuth的文章，阮老师的这篇是最好懂的，拜读了

2014年5月14日 08:19 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-323237) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**M16 说：**

说白了oauth就是一个网络版的usbkey

2014年5月17日 13:50 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-323850) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**洋子 说：**

支付宝链接已失效

2014年5月18日 22:56 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-324040) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**Nicole 说：**

非常感谢老师的分享，收益了。。

2014年5月20日 09:13 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-324236) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**匆匆过客 说：**

**引用Charles的发言：**

本文关于state参数的解释，是语焉不详的。事实上，绝大多数的互联网中文文档对这个参数都语焉不详。我想，对于造成近期互联网上危害广泛的OAuth漏洞来说，众多中文技术资料对这个参数解释不到位，对这个参数的实现没有给出清晰的指导，也是成因的一部分。原文是这么说的：RECOMMENDED. An opaque value used by the client to maintain state between the request and callback. The authorization server includes this value when redirecting the user-agent back to the client. The parameter SHOULD be used for preventing cross-site request forgery as described in Section 10.12.如果阮老师只是翻译为什么“客户端状态”之类的话，还有什么“可以随便填”，这算不上一种到位的解释。首先，这个参数是“RECOMMENDED”，并非什么可有可无的东西，事实是很多厂商都实现成了可有可无，其次，这个参数是“SHOULD”，为啥“SHOULD”呢，因为会引发“CSRF”。

仔细一看，确实，如果state项不用动态数据的话会存在CSRF漏洞

2014年5月20日 09:39 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-324238) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**smilexu 说：**

（E）客户端使用令牌，向资源服务器申请获取资源。

令牌就可以？整个过程都没有用户名和密码的获得，那资源服务器又是如何确认用户的？

2014年5月21日 12:01 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-324450) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**万文婷 说：**

我是比较文学与世界文学的硕士生，希望您能联系我，探讨一下卡尔维诺。

2014年5月21日 16:19 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-324482) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**niannian 说：**

授权码模式中，D步骤，文中说client\_id是必选项，但是随后的例子中没有这项

2014年5月23日 14:12 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-324809) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**spojus 说：**

收款主页下线啦

2014年5月24日 23:01 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-324975) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**loddit 说：**

同ls的niannian  
另外 granttype authorizationcode 都没加 \_ 下划线

2014年5月24日 23:20 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-324977) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**Ike 说：**

**引用phi的发言：**

那什么，没看懂。能不能用位图。ASCII图看的很不习惯。谢谢

話說，RFC本來就是純文字檔案，這樣的圖其實看了很親切。。。

2014年5月25日 16:14 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-325078) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**rhgb 说：**

**引用smilexu的发言：**

（E）客户端使用令牌，向资源服务器申请获取资源。

令牌就可以？整个过程都没有用户名和密码的获得，那资源服务器又是如何确认用户的？

使用oauth本来就是为了避免客户端获得用户名和密码…… 资源服务器如何用用户名密码来确认用户，就如何用令牌来确认用户。令牌就是一种包含（隐含）用户id的一次性的密码

2014年6月 7日 16:47 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-327535) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**king 说：**

如果客户端做一个假的验证授权页面，来套取用户，用户名和密码，是有可能的吧？

2014年6月12日 11:30 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-328221) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**jucelin**](http://jucelin.com/)**说：**

**引用king的发言：**

如果客户端做一个假的验证授权页面，来套取用户，用户名和密码，是有可能的吧？

我也有此疑问。weibo.com提供的都提示"请认准本页URL地址必须以 api.weibo.com 开头"，这提示基本上没用。

2014年6月16日 20:41 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-328876) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**mll 说：**

协议中的 SHOULD，应该等同于 MUST 来对待。

2014年6月19日 17:25 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-329069) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**sjh**](http://songjinghe.byethost7.com/)**说：**

**引用jucelin的发言：**

我也有此疑问。weibo.com提供的都提示"请认准本页URL地址必须以 api.weibo.com 开头"，这提示基本上没用。

工商银行的网银在开通时要求用户自己说一句话，每次使用网银付款时网站会向用户出示这句话以证明网站的合法性。这个感觉要好些吧~

2014年7月18日 17:27 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-331286) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**lvqiuyi 说：**

有两点不太明白  
1.授权码授权里，为什么要两次传递重定向url呢？有什么好处？  
2.D步骤里，传递的参数有Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW，什么作用呢？

2014年8月 5日 11:24 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-332867) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**lvqiuyi 说：**

懂了，Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW，应该是对于应用client的授权认证，这样就相当于有两层保护，一层是认证客户端，一层是认证用户，不过basic的加密方式，是不是不太安全呢？

2014年8月 5日 14:44 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-332891) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**suolaiwo 说：**

请问一下，授权码code，其作用是什么？

2014年9月 8日 14:29 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-336963) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**guest 说：**

**引用suolaiwo的发言：**

请问一下，授权码code，其作用是什么？

本质的作用是避免Access Token通过URL返回，有篇视频讲这个比较详细《[开放平台]一步一步理解OAuth2协议-全网首发》  
<http://edu.51cto.com/index.php?do=course&m=addlession&course_id=2035>

2014年9月16日 14:30 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-338157) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**zhiwen 说：**

请问你这个是使用什么框架实现的OAuth的相关服务器端的？

2014年9月18日 13:55 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-338437) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**陈伟**](http://pinbaijia.cn/)**说：**

正在学习OAuth2，写得太好了！非常容易理解！

2014年10月21日 16:57 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-341907) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**hebidu 说：**

我想知道 如何 生成access\_token ，refresh\_token 。

2014年10月22日 12:45 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-341942) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**James 说：**

**引用hebidu的发言：**

我想知道 如何生成access\_token ，refresh\_token 。

生成access\_token ，refresh\_token 很简单,取一唯一的随机信息再做BASE64编码即可,它本身不包含任何有效信息的;服务端把它和用户受权关联以校验其受权的有效性.

2014年10月29日 10:45 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-342264) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**惜今草 说：**

请问，OAuth2.0客户端认证模式怎么限制客户端到认证服务器申请权限的范围(Scope)？

2014年11月12日 16:11 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-343428) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**yxwoods 说：**

通俗易懂！感觉如等到风来吹开京都雾霾，重见蓝天白云！

2014年11月13日 09:25 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-343516) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**jinlong 说：**

引导用户到授权界面，用户会先登录后授权，整个授权过程中没有提到究竟是为“谁”授权的啊？ 那token是怎么和userid进行映射的呢？  
我开始理解的是，用户同意授权后，会告诉授权服务器client\_id等参数外还会有userid。

2014年11月21日 16:12 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-344513) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**HM 说：**

阮老师为啥只发文而不答复上面技术性的探讨呢？

2014年12月 9日 12:47 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-346024) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**feiniu5566 说：**

**引用jinlong的发言：**

引导用户到授权界面，用户会先登录后授权，整个授权过程中没有提到究竟是为“谁”授权的啊？ 那token是怎么和userid进行映射的呢？  
我开始理解的是，用户同意授权后，会告诉授权服务器client\_id等参数外还会有userid。

你要知道这个授权的目的是给第三方应用授权，让他有权利去获取用户放在服务器上的资源，所以你应该站在第三方应用的角度来看待这篇文章，你的目的就是获取用户  
QQ空间里的图片，就这么理解吧，差不想为那个什么讯发广告

2014年12月 9日 16:31 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-346077) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**yuandghn 说：**

**引用匆匆过客的发言：**

仔细一看，确实，如果state项不用动态数据的话会存在CSRF漏洞

确实是这样的，很多时候大家都忽略了这个state参数存在的意义。企鹅家的OAuth2.0文档里倒是把这个参数说的很详细(非广告，实话实说)。

2015年1月25日 18:00 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-347506) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**smithfox 说：**

密码模式的例子中Request Header 中的 Authorization 是怎么来的？

2015年1月28日 12:39 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-347587) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**xiaozi0lei**](http://www.tigerbull.info/)**说：**

**文章写的不错，深入浅出，正好在找Oauth 2.0相关的介绍文章，有帮助，多谢！**

2015年3月 3日 15:49 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-348327) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**alleme 说：**

**引用yuandghn的发言：**

确实是这样的，很多时候大家都忽略了这个state参数存在的意义。企鹅家的OAuth2.0文档里倒是把这个参数说的很详细(非广告，实话实说)。

确实,刚查了腾讯文档 "state 可选 client端的状态值。用于第三方应用防止CSRF攻击，成功授权后回调时会原样带回。"

2015年3月11日 11:05 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-348463) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**Shion 说：**

拿到访问令牌（access token）后，是不是每次请求资源都必须附上访问令牌？OAuth2是不是只适合Http，基于OAuth2的都只能通过客户端轮询来获取最新状态而不能由服务端进行推送？

2015年4月10日 16:38 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-349016) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**test 说：**

简化模式中：  
C）假设用户给予授权，认证服务器将用户导向客户端指定的"重定向URI"，并在URI的Hash部分包含了访问令牌。

（D）浏览器向资源服务器发出请求，其中不包括上一步收到的Hash值。

（E）资源服务器返回一个网页，其中包含的代码可以获取Hash值中的令牌。  
c处的hash有没有多余呢？谢谢

2015年4月14日 08:21 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-349078) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**许广辉 说：**

每次看到你这儿，东西就看明白了，简单清晰

2015年4月16日 15:52 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-349137) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**shengzhao**](http://shengzhaoli.com/)**说：**

具体的参考实现: <http://git.oschina.net/shengzhao/spring-oauth-server>

整合Spring Security与Oauth2 的完整代码.

2015年4月20日 15:00 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-349185) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**song 说：**

描述的很好，不错。多谢了！

2015年5月13日 17:46 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-349639) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**小正 说：**

首先谢谢阮老师，学习了，我想问下token存在数据库还是session里好呢  
存数据库是不是存一个过期时间(或者之类的)，还有如果用户操作了，这个token就延长时间还是固定死的(几个小时就几小时几天就几天)之后过期再用refresh取

2015年6月 3日 15:18 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-350107) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**小正 说：**

**引用小正的发言：**

首先谢谢阮老师，学习了，我想问下token存在数据库还是session里好呢  
存数据库是不是存一个过期时间(或者之类的)，还有如果用户操作了，这个token就延长时间还是固定死的(几个小时就几小时几天就几天)之后过期再用refresh取

我查了一些资料，基本捋顺了

2015年6月 4日 09:37 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-350119) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**杨鹏 说：**

感谢分享！

2015年7月18日 09:49 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-350904) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**小超人 说：**

授权码模式,为什么不能通过一次请求获取访问令牌?为什么非要加上一次授权码请求呢?现在授权码的作用是指定访问范围,那为什么不可以把访问范围作为参数传给认证服务器通过一次请求获取访问令牌呢?

2015年7月20日 10:09 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-350923) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**小超人 说：**

**引用小超人的发言：**

授权码模式,为什么不能通过一次请求获取访问令牌?为什么非要加上一次授权码请求呢?现在授权码的作用是指定访问范围,那为什么不可以把访问范围作为参数传给认证服务器通过一次请求获取访问令牌呢?

授权码模式比简化模式好在哪里呢?求路过大神指点

2015年7月20日 10:19 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-350924) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**小超人 说：**

**引用小超人的发言：**

授权码模式比简化模式好在哪里呢?求路过大神指点

是安全性高吗?安全性高在哪里呢?求举例说明...

2015年7月20日 10:23 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-350925) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**卡萨布兰卡**](http://www.dplor.com/)**说：**

有一处clientcredentials，应该改为：client\_credentials

2015年8月 3日 16:58 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-351108) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**newnius**](http://blog.newnius.com/)**说：**

**引用jinlong的发言：**

引导用户到授权界面，用户会先登录后授权，整个授权过程中没有提到究竟是为“谁”授权的啊？ 那token是怎么和userid进行映射的呢？  
我开始理解的是，用户同意授权后，会告诉授权服务器client\_id等参数外还会有userid。

这个token可以是惟一的，也就是说，token唯一确定一个用户，像@feiniu5566 说的，客户端可以通过令牌请求用户信息，包括userid，所以没有必要额外发送。

2015年8月13日 17:35 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-351247) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**williamwue 说：**

写得十分清楚明白，多谢博主！

2015年8月19日 16:20 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-351330) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**helm 说：**

阮老师，授权码模式中A步骤中的redirect\_uri写的是可选项，D步骤说的是必选项，这里是不是有矛盾

2015年8月22日 12:15 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-351369) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**damon 说：**

大学的时候接触淘宝API，也碰到：授权码模式， 但是知道看了这篇文章才系统的理解。。  
谢了！

2015年9月15日 18:01 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-351695) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**xiaoC**](http://yiwanbaqianli.com/)**说：**

最近在搞第三方登入，虽然已经按文档弄好了，但是并不懂OAuth为何物，看了楼主介绍总结，很是详细，多谢楼主

2015年9月20日 01:27 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-351794) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**我爱demo 说：**

最近要弄这个， 虽然几天了还是没什么进度，不过看了这篇文章感觉很详细，虽然并不懂。

2015年10月 6日 16:35 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-351993) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**aehyok 说：**

灰常好，学到了很多，谢谢大拿

2015年10月10日 15:22 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-352071) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**分崩离析**](http://www.baidu.com/)**说：**

**引用smilexu的发言：**

（E）客户端使用令牌，向资源服务器申请获取资源。

令牌就可以？整个过程都没有用户名和密码的获得，那资源服务器又是如何确认用户的？

令牌就可以代表用户，资源服务器只负责提供资源

2015年11月 4日 10:39 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-352570) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**陈柏成 说：**

（C）假设用户给予授权，认证服务器将用户导向客户端事先指定的"重定向URI"（redirection URI），同时附上一个授权码。

我不明白这个URI是用来跳转到认证服务器的url 还是指向前面例子说的云冲印的图片 谢谢了^^

2015年11月16日 11:40 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-352800) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**h**](http://ddd/)**说：**

**引用分崩离析的发言：**

令牌就可以代表用户，资源服务器只负责提供资源

这个令牌是一个字符串，资源服务器如何鉴别这个字符串不是伪造的呢？需要资源服务器到认证服务器上去鉴别吗，还是两边约定一种鉴别机制(比如鉴别RMB一样)

2015年12月16日 10:08 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-353333) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**h**](http://ddd/)**说：**

**引用h的发言：**

这个令牌是一个字符串，资源服务器如何鉴别这个字符串不是伪造的呢？需要资源服务器到认证服务器上去鉴别吗，还是两边约定一种鉴别机制(比如鉴别RMB一样)

看了 RFC 文档  
<http://tools.ietf.org/html/rfc6749>  
明白了，还是需要在 AS 和 RS 之间有个交互。

7. Accessing Protected Resources

The client accesses protected resources by presenting the access  
token to the resource server. The resource server MUST validate the  
access token and ensure that it has not expired and that its scope  
covers the requested resource. The methods used by the resource  
server to validate the access token (as well as any error responses)  
are beyond the scope of this specification but generally involve an  
interaction or coordination between the resource server and the  
authorization server.

2015年12月16日 11:42 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-353336) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**h**](http://kkk/)**说：**

**引用h的发言：**

这个令牌是一个字符串，资源服务器如何鉴别这个字符串不是伪造的呢？需要资源服务器到认证服务器上去鉴别吗，还是两边约定一种鉴别机制(比如鉴别RMB一样)

<http://tools.ietf.org/html/rfc6749>

Accessing Protected Resources

The client accesses protected resources by presenting the access  
token to the resource server. The resource server MUST validate the  
access token and ensure that it has not expired and that its scope  
covers the requested resource. The methods used by the resource  
server to validate the access token (as well as any error responses)  
are beyond the scope of this specification but generally involve an  
interaction or coordination between the resource server and the  
authorization server.

少不了 RS 到 AS 之间交互认证 access\_token，这个 Oauth 没有规定啊,可以随意发挥啊。

2015年12月16日 11:45 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-353337) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**兴杰 说：**

看了好几次，今天终于明白第一种模式了，谢谢老师 ^^

2015年12月18日 19:00 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-353382) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**Ans 说：**

不错，好文章！

2015年12月23日 11:01 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-353472) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**Laughing\_Lz 说：**

restful一头雾水，老师太深奥，还是不懂，泪奔啊

2015年12月23日 17:56 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-353483) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**喵喵 说：**

**引用陈柏成的发言：**

（C）假设用户给予授权，认证服务器将用户导向客户端事先指定的"重定向URI"（redirection URI），同时附上一个授权码。

我不明白这个URI是用来跳转到认证服务器的url 还是指向前面例子说的云冲印的图片谢谢了^^

AS完成授权之后 把授权码和state附在重定向URI之后 再把浏览器重定向回这个地址 redirection URI是client提供的一个地址 做这一步是为了让Client能够获得授权码

2015年12月24日 10:25 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-353492) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**喵喵 说：**

请教阮老师和路过大神：

我一直对implicit grant中的web-hosted client resource的作用感到疑惑  
为什么需要先向web-hosted client resource请求一个包含JS代码html 然后再用JS来读取access\_token in Fragment(Hash)最后获得access\_token 为什么不直接在本地用JS代码来读取呢？ 我在stackoverflow上看到人说这是重定向的常用做法 并没有涉及到安全性的考虑 但是为什么要这样做呢？

2015年12月24日 10:36 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-353493) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**喵喵 说：**

**引用小超人的发言：**

授权码模式,为什么不能通过一次请求获取访问令牌?为什么非要加上一次授权码请求呢?现在授权码的作用是指定访问范围,那为什么不可以把访问范围作为参数传给认证服务器通过一次请求获取访问令牌呢?

**引用小超人的发言：**

授权码模式比简化模式好在哪里呢?求路过大神指点

恩 先说为什么implicit grant不需要Authorization code吧 首先因为Authorization code和client secret是不会暴露给用户的 这个是服务器与服务器之间通信才需要的 所以implicit grant是通过user-agent（web browser）来与AS交互 你加上这两个东西之后 还是会暴露给用户看到的 所以没有必要在简化模式中增加他的复杂性

现在回到你问的 为什么授权码模式需要这个授权码 当然是为了安全性 首先在OAuth体系中access\_token是作为访问获取资源的唯一凭据 如果在AS授权完成之后 直接通过重定向传回access\_token 那么HTTP 302不是安全的 Attacker有可能会获取到access\_token 但是如果只返回Authorization code 就算别人获得了也没什么卵用 因为Authorization code不能获取到资源 在client向AS请求access\_token的过程中 是通过HTTPS来保证安全的 而且获得access\_token是需要client secret与Authorization code一起的 Attacker知道了Authorization code但并不知道client secret 同样也不能获得到access\_token 所以client与AS是有责任保护好client secret的

获得了access\_token之后 向RS发起请求 RS其实会与AS交互 来校验access\_token 所以你想直接伪造一个access\_token 那也是不ok的

2015年12月24日 11:03 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-353496) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**存在感为零的人 说：**

很透彻

2015年12月24日 11:59 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-353498) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**喵喵 说：**

突然想到漏说了一块 关于为什么不直接用HTTPS重定向回client  
是因为不是所有client server都支持HTTPS~所以为了通用性 和安全性 才衍生出来这么一个Auth code

但是AS肯定是实现HTTPS的 所以在client向AS提起request 是木有问题的~

2015年12月24日 14:54 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-353502) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**李先森**](http://www.baidu.com/)**说：**

不太明白 为什么使用auth\_code换取access\_token的时候 还要传redirect\_uri

2016年1月10日 13:39 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-353913) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**北言 说：**

最近在学习oauth2,看了阮老师的文章,很透彻,明白了.

2016年2月17日 11:33 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-354535) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**看法 说：**

写的很好，有参考价值

2016年2月22日 17:37 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-354621) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**雅爱 说：**

您好，想谈论下，在更新资源令牌的时候为什么要使用专门的更新令牌来更新资源令牌？为何不直接用资源令牌来更新资源令牌？

2016年3月25日 08:51 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-355249) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**popwar 说：**

有些解释不够详细，看看这个tutorial更容易理解http://tutorials.jenkov.com/oauth2/index.html

2016年4月 1日 13:40 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-355374) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**zsxkarl 说：**

最近在了解oauth，多谢详细讲解！

2016年4月26日 21:11 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-355906) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**Ming 说：**

阮老师的文章还是非常好的。

2016年6月15日 15:27 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-358472) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**Fishy 说：**

阮老师，请问一下。  
密码模式里的B步骤里  
提到的Authorization: Basic 码是如何获得的呢？  
在网上查到是userName：password的形式以Base64编码得到的，但是实际操作中得到的结果不一致。  
所以想问下，该值是怎么编码得出的呢？

2016年6月17日 09:19 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-359282) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**Fishy 说：**

阮老师，请问一下。  
密码模式里的B步骤里  
提到的Authorization: Basic 码是如何获得的呢？  
在网上查到是userName：password的形式以Base64编码得到的，但是实际操作中得到的结果不一致。  
所以想问下，该值是怎么编码得出的呢？

2016年6月17日 10:10 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-359286) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**robottech 说：**

为什么用户授权客户端访问自己的资源后，授权服务器给客户端返回一个code 然后客户端通过code再向授权服务器申请令牌 然后拿着令牌去访问资源 而不是 用户授权客户端访问自己的资源后 授权服务器直接返回令牌给客户端？为啥要费个二道手，先返回CODE呢

2016年7月 8日 10:17 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-360355) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**Euclidvi31 说：**

client secret 这个概念漏掉了吧

2016年7月31日 21:38 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-361479) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**路人甲**](http://meiyou/)**说：**

**引用niannian的发言：**

授权码模式中，D步骤，文中说client\_id是必选项，但是随后的例子中没有这项

client\_id的确应该是必选项，应该是阮老师遗漏了

2016年8月 8日 15:17 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-361972) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**路人乙**](http://meiyou/)**说：**

**引用jucelin的发言：**

我也有此疑问。weibo.com提供的都提示"请认准本页URL地址必须以 api.weibo.com 开头"，这提示基本上没用。

看新浪，腾讯的OAuth方案，客户端接入是需要申请，经过服务提供商审核的（会发放 client\_id, cilent\_secret），如果假冒一个钓鱼网站的话，审核是很有可能不通过的，甚至自己坑自己。  
（当然审核可能不一定100%杜绝）

2016年8月 8日 15:25 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-361973) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**路人丙**](http://meiyou/)**说：**

**引用Euclidvi31的发言：**

client secret 这个概念漏掉了吧

client secret是必须的，阮老师遗漏了

2016年8月 8日 15:31 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-361974) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**路人丁**](http://meiyou/)**说：**

**引用李先森的发言：**

不太明白 为什么使用auth\_code换取access\_token的时候 还要传redirect\_uri

redirect\_uri是回调地址，客户端需要接受access\_token，如果不跟认证服务器说好，认证服务器怎么给呢？

2016年8月 8日 15:34 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-361976) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**路任戊**](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)**说：**

**引用路人乙的发言：**

看新浪，腾讯的OAuth方案，客户端接入是需要申请，经过服务提供商审核的（会发放 client\_id, cilent\_secret），如果假冒一个钓鱼网站的话，审核是很有可能不通过的，甚至自己坑自己。  
（当然审核可能不一定100%杜绝）

楼上好像误解了别人的问题，jucelin的意思是一个网站提供例如新浪微博登陆，在引导用户到新浪认证时，其实时引导到一个虚假的地址。所以审核没用啊。  
我觉得OAuth2.0对这个问题无解，因为如果既然会构造虚假地址，证明客户端就是一个危险站点。他能够引诱用户访问基本就成功了

2016年8月 8日 18:56 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-361991) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**Jim 说：**

**引用popwar的发言：**

有些解释不够详细，看看这个tutorial更容易理解http://tutorials.jenkov.com/oauth2/index.html

看了这个链接，理解更加深刻!

2016年8月24日 15:55 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-363046) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**qihaiyan 说：**

（D）浏览器向资源服务器发出请求，其中不包括上一步收到的Hash值。

（E）资源服务器返回一个网页，其中包含的代码可以获取Hash值中的令牌。

implicit模式中这2行解释描述好像有问题，按照图中的说明，“浏览器向资源服务器发出请求”，应该是浏览器向客户端的web服务器发出请求，而不是向资源服务器发出请求。

2016年9月 4日 12:50 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-363772) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**大庆 说：**

阮老师能不能形象的讲一下 OAuth，AspNetIdentity，OWin，IdentityServer3这四个的关系？以及OAuth的四种模式分别适合在什么场景下使用？一直比较迷糊，求解惑，谢谢

2016年9月14日 00:27 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-364449) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**tony**](http://www.lihancong.cn/)**说：**

很清晰, 有参考价值

2016年9月28日 21:25 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-365376) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**Rabon**](http://rabondai.github.io/)**说：**

真是简洁明了，通俗易懂啊，谢谢博主分享。

2016年10月26日 15:40 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-367130) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**ww 说：**

一口气看完，第一感觉是，恩，似乎明白了，OAuth大概就是这样的东西啊！  
但是回顾一遍，很多细节还是不明所以，而且持续想不通。

1、在第六章，授权码模式（authorization code）中，  
先从通过“用户授权操作”得到了“认证码（Auth Code）”，  
然后再用“认证码（Auth Code）”换取到了“令牌（Token）”，  
为什么要分2步呢？直接通过“授权”取得“令牌”不就行了？

2、取到“令牌（Token）”的用途，是用它来“识别用户”并获取信息，  
而且令牌又基本是明文传输的，  
假设我无意中看到了其他用户的令牌，是不是可以直接拿来用呢？  
难道不担心令牌被“冒用”吗？

求解惑。

2016年12月 2日 16:37 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-369616) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**Pumpkin 说：**

可参考：  
<http://bitoftech.net/2014/06/01/token-based-authentication-asp-net-web-api-2-owin-asp-net-identity/>

2016年12月 7日 06:11 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-369933) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**鱼肚**](http://yudu.tech/)**说：**

（D）客户端收到授权码，附上早先的"重定向URI"，向认证服务器申请令牌。这一步是在客户端的后台的服务器上完成的，对用户不可见。

在 【六、授权码模式】 中的这段翻译有误，原文中是说附上早先的授权码。这一步设置的重定向URI不一定非要和早先的相一致。

2016年12月 7日 11:45 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-369978) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**鱼肚**](http://yudu.tech/)**说：**

@ww：

关于为什么要多一步授权码的解释：http://stackoverflow.com/questions/13387698/why-is-there-an-authorization-code-flow-in-oauth2-when-implicit-flow-works-s

简单来说，是因为HTTP重定向没有body，只能通过url传参数，而url中的参数是不安全的，因为所有经过的路由器或服务器都能读取到url的信息，所谓的中间人攻击。

2016年12月 7日 11:50 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-369982) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**鱼肚**](http://yudu.tech/)**说：**

**引用鱼肚的发言：**

（D）客户端收到授权码，附上早先的"重定向URI"，向认证服务器申请令牌。这一步是在客户端的后台的服务器上完成的，对用户不可见。

在 【六、授权码模式】 中的这段翻译有误，原文中是说附上早先的授权码。这一步设置的重定向URI不一定非要和早先的相一致。

自己给自己打脸。关于这段是我之前理解有误，请求access\_token时确实需要早先的redirect\_uri,以便服务器校验code和redirect\_uri是否是一对

2016年12月16日 10:34 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-370469) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**Terence Mak**](http://terence-mak.blogspot.com/)**说：**

很感謝你的阮大大的文章，看了好多網路上的文章沒有看得太懂，看完你的終於明白了，特意說聲謝謝 :)

2017年2月 2日 15:59 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-372332) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**Arya**](http://aryablog.xyz/)**说：**

在研究 Google API 的时候，关于授权部分有一些疑惑，感谢您的博文。

2017年2月 5日 14:33 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-372370) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**清风 说：**

第六节 授权码模式，D步骤的例子中没有client\_id参数，与描述中的“必选项”矛盾？

2017年2月10日 16:11 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-372669) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**grayVTouch**](http://www.lysqdx.com/drugInterdiction/index.php)**说：**

每一次不小心浏览到阮大神的博客，都会被其提供的干货好文吸引！这样的文章越多越好，支持 阮sir

2017年3月11日 14:53 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-374499) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**etata**](http://www.cnblogs.com/etata/)**说：**

简单来说就是

在第三方服务那里， 加一个链接，链到服务器，在服务器输入账号密码之后， 服务器返给你一个 字符串 也就是授权码或者token之类都可以。然后拿着第三方服务拿着这个授权码 和服务器进行通信。  
这个保护服务器上的账号密码，避免被过多的曝光，减少风险。

2017年3月14日 09:44 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-374691) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**小鱼 说：**

讲句真的，我估计我蠢笨啊，愣是觉得看不懂，还有没看明白为啥

2017年3月23日 12:57 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-375325) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**xwei 说：**

@鱼肚：

而且code就可以不需要保密了，因为code是一次性的，只要用过一次就失效了，即使中间人拿到code也不能获取access\_token，因为还需要client\_id和client\_secret，而这些信息是在资源服务器保存的。

2017年3月26日 11:15 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-375461) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**gink 说：**

**引用newnius的发言：**

这个token可以是惟一的，也就是说，token唯一确定一个用户，像@feiniu5566 说的，客户端可以通过令牌请求用户信息，包括userid，所以没有必要额外发送。

token 是第三方服务器向授权服务器用 code 获取的,那么授权服务器是怎么知道这个code是哪个用户授权的？  
code也是唯一的，而且关联了用户？如果是这样，应该就不存在什么CSRF攻击

2017年4月18日 10:25 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-376408) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**normalhefei 说：**

想问一下授权码模式中，为什么要有一个授权码？多一个来回？ 为什么不直接返回一个token？

2017年4月27日 12:40 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-376715) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**KS 说：**

**引用normalhefei的发言：**

想问一下授权码模式中，为什么要有一个授权码？多一个来回？ 为什么不直接返回一个token？

想真正了解OAUTH得多看看，这篇文章说是流水账也不为过，讲了一堆，只能让人了解了What和省略了很多细节How，至于评论中的一大堆Why就没有说明了。感觉典型的挖坑文章，挖完坑不填有时候还不如不挖。

ok，回到你的问题：服务器返回的授权码是一个客户端用于请求真正的accessToken的ticket。  
为什么这么干：  
因为服务器此时只是完成了：确认用户授权，但是至于用户确认授权的是不是真正clientId代表的app还不确定？  
因为clientId那玩意公开的，用户都知道，所以如果我知道了clientId，我可以说我是你。  
所以不安全，要做第二次验证。 第二次验证的就是在客户端后台完成的了，它得把clientId连同clientCredentail（secrectKey或者说密码）再加上获得的AuthorizationCode发给服务器做验证。  
这才能获取最终使用的accessToken。

2017年5月 1日 11:18 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-376803) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**KS 说：**

@ww：

第一个问题估计你知道答案了。

第二个问题是一个good point,答案是：所以不能用http，只能用https。参见 RFC6749:https://tools.ietf.org/html/rfc6749  
Since requests to the authorization endpoint result in user  
authentication and the transmission of clear-text credentials (in the  
HTTP response), the authorization server MUST require the use of TLS  
as described in Section 1.6 when sending requests to the  
authorization endpoint.

2017年5月 1日 11:34 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-376804) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**Victor**](http://xn--8ov87j/)**说：**

看了这个文档，懂得了什么是auth2，但是对于JAVA中如何能够应用到项目中呢？

2017年5月 2日 21:42 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-376840) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**gs 说：**

非常好的文章。

2017年6月16日 17:54 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-378015) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**刘伟**](http://www.cqkd6381.com/)**说：**

有看不明白的可以去慕课网上看看这个课程，更容易明白  
<http://www.imooc.com/learn/557>

2017年7月 5日 09:43 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-378480) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**w666 说：**

在“客户端模式”和“更新令牌”章节中，我发现写的是"grantType","clientcredentials","refreshtoken",看代码结果是"grant\_type","client\_credentials","refresh\_token"，这是作者不小心写错了还是说是一种书写约定，请问

2017年7月 7日 11:42 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-378551) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**cdy 说：**

redirecturl在授权码模式中a和e中可以不一样的吧，这两个地方的 url？

2017年7月19日 09:07 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-378772) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**Kolin 说：**

完全没有任何头绪，一点都看不懂

2017年8月 4日 14:15 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-379238) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**Lwr 说：**

意思是懂了，具体怎么实现的还需要多加学习。

2017年8月 8日 11:50 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-379338) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**jast 说：**

请问：  
七、简化模式  
简化模式（implicit grant type）不通过第三方应用程序的服务器，直接在浏览器中向认证服务器申请令牌，跳过了"授权码"这个步骤，因此得名。所有步骤在浏览器中完成，令牌对访问者是可见的，且客户端不需要认证。  
中的：

（B）用户决定是否给于客户端授权。

这个用户怎么决定？用户是否需要登入？

还有：这种模式需不需要第三方应用程序的服务器登入用户（即输入用户名密码）？

2017年8月 8日 16:54 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-379346) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**Chova**](http://www.weillp0.com/)**说：**

**引用路任戊的发言：**

楼上好像误解了别人的问题，jucelin的意思是一个网站提供例如新浪微博登陆，在引导用户到新浪认证时，其实时引导到一个虚假的地址。所以审核没用啊。  
我觉得OAuth2.0对这个问题无解，因为如果既然会构造虚假地址，证明客户端就是一个危险站点。他能够引诱用户访问基本就成功了

没看见state这个参数吗？这个就是防止虚假链接的

2017年8月18日 11:41 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-379632) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**兰兰同学 说：**

该篇文章里的authorization server翻译成中文难道不应该是授权服务器吗？认证服务器应该是authentication server才对吧？

2017年9月 6日 14:57 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-380088) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**Chappie 说：**

**引用qihaiyan的发言：**

（D）浏览器向资源服务器发出请求，其中不包括上一步收到的Hash值。

（E）资源服务器返回一个网页，其中包含的代码可以获取Hash值中的令牌。

implicit模式中这2行解释描述好像有问题，按照图中的说明，“浏览器向资源服务器发出请求”，应该是浏览器向客户端的web服务器发出请求，而不是向资源服务器发出请求。

万分感谢，看了好久没看明白，看到你的解释终于明白了，应该是向客户端web服务器发出请求，获取从urlHash的js应该客户端做处理

2017年9月11日 18:18 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-380266) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**丘彪 说：**

谢谢阮老师，清晰讲解。

2017年9月12日 11:30 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-380302) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**lll 说：**

写的也没多清晰 里面那么多回调uri, 都没说清楚到底 都是回调哪儿的

2017年9月18日 15:56 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-380575) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**Chuck**](https://blog.fedt.xin/)**说：**

一堆的用户，客户端等别名让这篇文章看起来非常费劲。

2017年10月23日 00:19 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-381568) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**菠萝 说：**

（A）用户访问客户端，后者将前者导向认证服务器。  
（B）用户选择是否给予客户端授权。  
（C）假设用户给予授权，认证服务器将用户导向客户端事先指定的"重定向URI"（redirection URI），同时附上一个token令牌。  
（D）客户端收到token令牌，向认证服务器发起查询请求。这一步是在客户端的后台的服务器上完成的，对用户不可见。  
（E）认证服务器核对了token令牌，确认有效期内后，向客户端返回用户信息

请问老师，这是哪一种模式啊？ 电信的天翼帐号开放平台好像就是这个流程。

2017年10月23日 17:17 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-381604) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**大新哥 说：**

您好，各位大神们，（搭建OAuth2.0标准的授权服务器+REST资源服务器，并提供用户注册服务和REST资源访问接口。）这个怎么去实现啊

2017年10月24日 15:54 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-381632) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**smallriver 说：**

@喵喵：

这个回答为啥需要授权码而不是直接返回access\_token基本是正确的，但是我觉得有两点可能不太对。第一点是说授权服务器返回302带上授权码是http的，这个不对，因为请求授权的接口本身可以是https的，返回302自然也就不会被窃取看到。第二点是用授权码获取access\_token接口不一定强制要求是https的，有clientSecret做加密足以区分非法用户。

2017年10月30日 22:43 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-381824) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**smallriver 说：**

**引用KS的发言：**

想真正了解OAUTH得多看看，这篇文章说是流水账也不为过，讲了一堆，只能让人了解了What和省略了很多细节How，至于评论中的一大堆Why就没有说明了。感觉典型的挖坑文章，挖完坑不填有时候还不如不挖。

ok，回到你的问题：服务器返回的授权码是一个客户端用于请求真正的accessToken的ticket。  
为什么这么干：  
因为服务器此时只是完成了：确认用户授权，但是至于用户确认授权的是不是真正clientId代表的app还不确定？  
因为clientId那玩意公开的，用户都知道，所以如果我知道了clientId，我可以说我是你。  
所以不安全，要做第二次验证。 第二次验证的就是在客户端后台完成的了，它得把clientId连同clientCredentail（secrectKey或者说密码）再加上获得的AuthorizationCode发给服务器做验证。  
这才能获取最终使用的accessToken。

KS说的挺好的，这篇文章属于总体上告诉你oauth2.0有哪几个步骤，有个基本认识而已。至于细节，还需要去看原文档。

关于为啥要引入授权码这一步，刚开始我也没太理解，看了KS的解释才瞬间懂了。就是说，如果没有授权码这一步，授权服务器在用户授权后直接返回accessToken，那会发生什么呢？就是随便一个客户端都可以把用户引导到授权页面，然后拿到accessToken去资源服务器玩了。因此，为了防止这种情况发生，在发放accessToken时，必须确定该客户端是在授权服务器这边注册过的（一般注册过会发放clientId&clientSecret），所以引入授权码这步，让客户端带上双方约定好的clientSecret，确认无误再发放accessToken，这才保证了安全。

2017年10月30日 23:18 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-381825) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**张三 说：**

为啥要用accessToken去获取clientId和userId 然后再用clientId和userId去调被授权服务器接口 我直接用accessToken不是也可以吗？没看懂这一步的作用

2017年11月13日 18:07 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-382281) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**l33klin**](http://www.klinlee.com/)**说：**

讲得通俗易懂，谢谢博主！

2017年11月16日 11:53 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-382378) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**jason 说：**

我有几个问题，关于“隐式授权”的。

1. 既然授权服务器已经把access token（不是授权token）发给agent了，为什么agent非要去取一个script解析access token，然后才能转发给client呢？授权服务器是故意给了一个“间接的”access token吗？这样设计的好处是什么呢？为什么没有直接给一个完整的access token，然后agent直接把token发给client不就行了吗？

2. 根据6749，implicit方式中，script是取自“web-hosted client resource”，我认为，既然是“web-hosted”，那么就不应该是“client”自己了。client有可能是本地应用。这个理解对吗？那个解析的script，为什么是“client”这边提供，而不是授权服务器这边提供呢？如果是client提供，那么完全可以把服务器发过来的完整的access token发过去，script的执行直接在client上就可以了，为什么非要取回到agent里面去执行呢？

oauth 2.0协议，感觉第一种实现方式非常严密，这种隐式的实现，总是感觉不够严密，但是不能理解为什么这样设计。

再次感谢提供这么好的解析，提前谢过您对我问题的答疑。

谢谢。

2017年11月30日 11:03 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-382844) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**李雨泽 说：**

请问一个问题，假如我现在有个门户网站，然后集成了一个论坛子系统，包括了论坛客户端以及论坛api服务，现在情况是论坛的客户端想调用论坛的api服务，那么就需要进行用户的身份验证以及论坛的客户端的身份验证。论坛客户端在这里能否理解成第三方应用？？？这种情况是否也适合使用oauth2?或者说使用于哪一种授权模式？期待你的回复

2018年1月 3日 18:37 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-384024) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**阿蒙 说：**

**引用兰兰同学的发言：**

该篇文章里的authorization server翻译成中文难道不应该是授权服务器吗？认证服务器应该是authentication server才对吧？

读的时候也发现这个问题了，看了看评论，果然有前辈提到这点。原文中也是一会儿“认证服务器”，一会儿“授权服务器”的。信息安全中“认证”和“授权”区分的贼清楚。

2018年1月19日 22:21 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-384763) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**Rayleigh 说：**

怎么实现在token有效期内，client可以在user第二次连接时候，避免再次授权。

2018年1月20日 12:31 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-384779) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**adi 说：**

access\_token由谁负责写入用户浏览器cookie中的

2018年2月 4日 09:54 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-385490) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**adi 说：**

**引用Rayleigh的发言：**

怎么实现在token有效期内，client可以在user第二次连接时候，避免再次授权。

token已经写入到用户浏览器的cookie了，当用户第二次访问连接子系统的时候，会将该cookie带上来。

2018年2月 4日 09:56 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-385491) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**落舞者**](http://1thx.com/)**说：**

根据 <https://tools.ietf.org/html/rfc6749#page-17> 协议 发现其中 更新令牌（Refresh Tokens） 章节 与 rfc6749描述 稍有出入  
其中例子 中多带两个参数client\_id和client\_secret 本文中没有涉及  
POST /token HTTP/1.1  
Host: server.example.com  
Content-Type: application/x-www-form-urlencoded

grant\_type=refresh\_token&refresh\_token=tGzv3JOkF0XG5Qx2TlKWIA  
&client\_id=s6BhdRkqt3&client\_secret=7Fjfp0ZBr1KtDRbnfVdmIw

rfc6749具体描述：

Refresh tokens MUST be kept confidential in transit and storage, and  
shared only among the authorization server and the client to whom the  
refresh tokens were issued. The authorization server MUST maintain  
the binding between a refresh token and the client to whom it was  
issued. Refresh tokens MUST only be transmitted using TLS as  
described in Section 1.6 with server authentication as defined by  
[RFC2818].

<https://tools.ietf.org/html/rfc6749#page-55>

2018年2月 6日 19:19 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-385604) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**F**](http://iforl.com/)**说：**

为什么将 implicit grant type 翻译为“简化模式”？理解为“隐式模式”不是更贴切吗？是不是将  
implicit 看成 simplicity 了？

2018年2月24日 16:46 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-385920) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**Bradley 说：**

老师有一点不明：  
（A）用户打开客户端以后，客户端要求用户给予授权。

（B）用户同意给予客户端授权。  
接着您又强调了关键在于B，即用户如何给予客户端授权  
但是接下来你并没有介绍这一块，而是直接跨过去从客户端如何用URI请求授权码开始了。。。

2018年3月 5日 15:46 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-386362) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**primary**](http://blog.csdn.net/Primary_wind)**说：**

赞！简洁易懂，现在各大开放平台采用的都是oauth2.0授权码模式!

2018年3月13日 20:13 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-386684) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**我的大名 说：**

**引用Bradley的发言：**

老师有一点不明：  
（A）用户打开客户端以后，客户端要求用户给予授权。

（B）用户同意给予客户端授权。  
接着您又强调了关键在于B，即用户如何给予客户端授权  
但是接下来你并没有介绍这一块，而是直接跨过去从客户端如何用URI请求授权码开始了。。。

授权码由认证服务器生成，客户端将通过浏览器跳转将用户导向认证服务器相关URI地址并在URI地址上附带客户端相关参数和redirect\_uri，在用户登录选择相关授权后,认证服务器通过浏览器跳转将用户导向redirect\_uri,并附带上授权码，浏览器访问这个uri，这个uri指向客户端的服务器，客户端服务器拦截浏览器请求获取到授权码。

2018年3月16日 14:42 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-386744) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**heropoo**](http://www.ioio.pw/)**说：**

赞！写的很详细

2018年4月 9日 11:09 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-387739) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**toccata 说：**

看了其他介绍oauth2.0的10篇文章，不如大牛的一篇文章。

看完豁然开朗。而之前看的其他几篇博客文章，我完全没发现他们讲的是同一个东西。

2018年4月19日 15:34 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-388110) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**sowork 说：**

**引用李先森的发言：**

不太明白 为什么使用auth\_code换取access\_token的时候 还要传redirect\_uri

可能是判断auth\_code真伪需要这些参数

2018年4月26日 14:25 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-388289) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**sowork 说：**

简化模式defg操作可以省略

2018年4月26日 16:17 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-388293) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**sowork 说：**

**引用sowork的发言：**

简化模式defg操作可以省略

或者说通常情况下不需要，如果想实现类似微信jssdk的功能，就和上面的流程对上了！！！

2018年4月26日 16:31 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-388294) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**沈林楠 说：**

知道了上面这些名词，就不难理解，OAuth的作用就是让"客户端"安全可控地获取"用户"的授权，与"服务商提供商"进行互动。  
这里的"服务商提供商"多了一个商字

2018年4月27日 16:43 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-388324) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**matt 说：**

太感謝分享了！！

2018年5月17日 13:21 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-388960) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**abc 说：**

请问OAuth1 和 OAuth2 区别。 怎么选择?

2018年6月 6日 15:11 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-389442) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**ring**](http://www.zixisi.cn/)**说：**

authorization server 更严谨的说法，应该是：授权服务器

2018年6月20日 15:36 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-389935) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**romanliu 说：**

这篇文章太过程式化，建议读一读这篇 <https://clickhelp.co/clickhelp-technical-writing-blog/how-does-single-sign-on-work-guide/>

2018年6月29日 16:40 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-390372) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

[**jack\_chen**](http://www.deepl.club/)**说：**

这些理解都是怎么来的呀。

2018年9月14日 15:51 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-393149) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**RexFang 说：**

应用场景章节部分，应该是来自官方 Example 里面的 README 吧？

2018年11月 3日 16:41 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-394685) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)

**唐 说：**

非常好，感恩，从一个传统业务程序，忽然做这一块，对于oauth等技术不了解。花了2小时通读查阅，谢谢！

2018年11月 4日 18:45 | [#](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-394713) | [引用](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html#comment-text)