#OAuth 2.0 文档

###初见OAuth2.0

OAuth 2.0是一个业界标准的授权协议，其定义了四种可以适用于各种应用场景的授权交互模式：授权码模式、应用授信模式、用户授信模式、简化模式。其中，授权码模式被广泛应用于第三方互联网开放平台，通过第三方登录是其最常见应用场景之一，比如使用微信、QQ和淘宝账号进行登录。

OAuth 2.0 官方文档地址：https://tools.ietf.org/html/rfc6749#section-4.3.2

## 一、情景再现：



## 二、令牌与密码

令牌（access\_token）与密码（password）的作用是一样的，都可以进入系统，但是有三点差异。

（1）令牌是短期的，到期会自动失效，用户自己无法修改。密码一般长期有效，用户不修改，就不会发生变化。

（2）令牌可以被数据所有者撤销，会立即失效。以上例而言，屋主可以随时取消快递员的令牌。密码一般不允许被他人撤销。

（3）令牌有权限范围（scope），比如只能进小区的二号门。对于网络服务来说，只读令牌就比读写令牌更安全。密码一般是完整权限。

上面这些设计，保证了令牌既可以让第三方应用获得权限，同时又随时可控，不会危及系统安全。这就是 OAuth 2.0 的优点。

注意，只要知道了令牌，就能进入系统。系统一般不会再次确认身份，所以**令牌必须保密，泄漏令牌与泄漏密码的后果是一样的。** 这也是为什么令牌的有效期，一般都设置得很短的原因。

## 三、互联网应用案例

### 情景再现：

简书，此网站为了方便用户登陆，他们支持很多个第三方登录，就拿微博登陆来说，用户在使用简书的网站的时候，为了方便登陆。他们会选择第三方登录，如果用户手机中有微博应用的话，弹出来一个授权按钮之后，用户点击授权，用户授权之后会生成一个授权码之类的字符串，简书服务器拿着这个串去访问用户的微博信息。例如微博昵称，微博的设置的地址，还可以访问微博绑定的手机号码，对于一些比较新的应用网站来时，通过支持第三方登录可以获取不少的用户量。

### 名词定义：

在详细讲解OAuth 2.0之前，需要了解几个专用名词。

（1） Client：准备访问用户资源的应用程序，其可能是一个web应用，或是一个后端web服务应用，或是一个移动端应用，也或是一个桌面可执行程序。  
（2）Resource Owner：顾名思义，资源的所有者，很多时候其就是我们普通的自然人（但不限于自然人，如某些应用程序也会创建资源），拥有资源的所有权。  
（3）User Agent：用户代理，本文中就是指浏览器。  
（4）Authorization server：授权服务器，在获取用户的同意授权后，颁发访问令牌给应用程序，以便其获取用户资源。  
（5）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。

### 实现思路：

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"，通过登录授权层，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，可以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。

### 运行流程

OAuth 2.0的运行流程，摘自RFC 6749。

+--------+ +---------------+  
 | |--(A)- Authorization Request ->| Resource |  
 | | | Owner |  
 | |<-(B)-- Authorization Grant ---| |  
 | | +---------------+  
 | |  
 | | +---------------+  
 | |--(C)-- Authorization Grant -->| Authorization |  
 | Client | | Server |  
 | |<-(D)----- Access Token -------| |  
 | | +---------------+  
 | |  
 | | +---------------+  
 | |--(E)----- Access Token ------>| Resource |  
 | | | Server |  
 | |<-(F)--- Protected Resource ---| |  
 +--------+ +---------------+  
   
  
（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。

## 五、OAuth授权的四种方式

官方文档关于OAuth的定义：https://tools.ietf.org/html/rfc6749#section-4.3.2

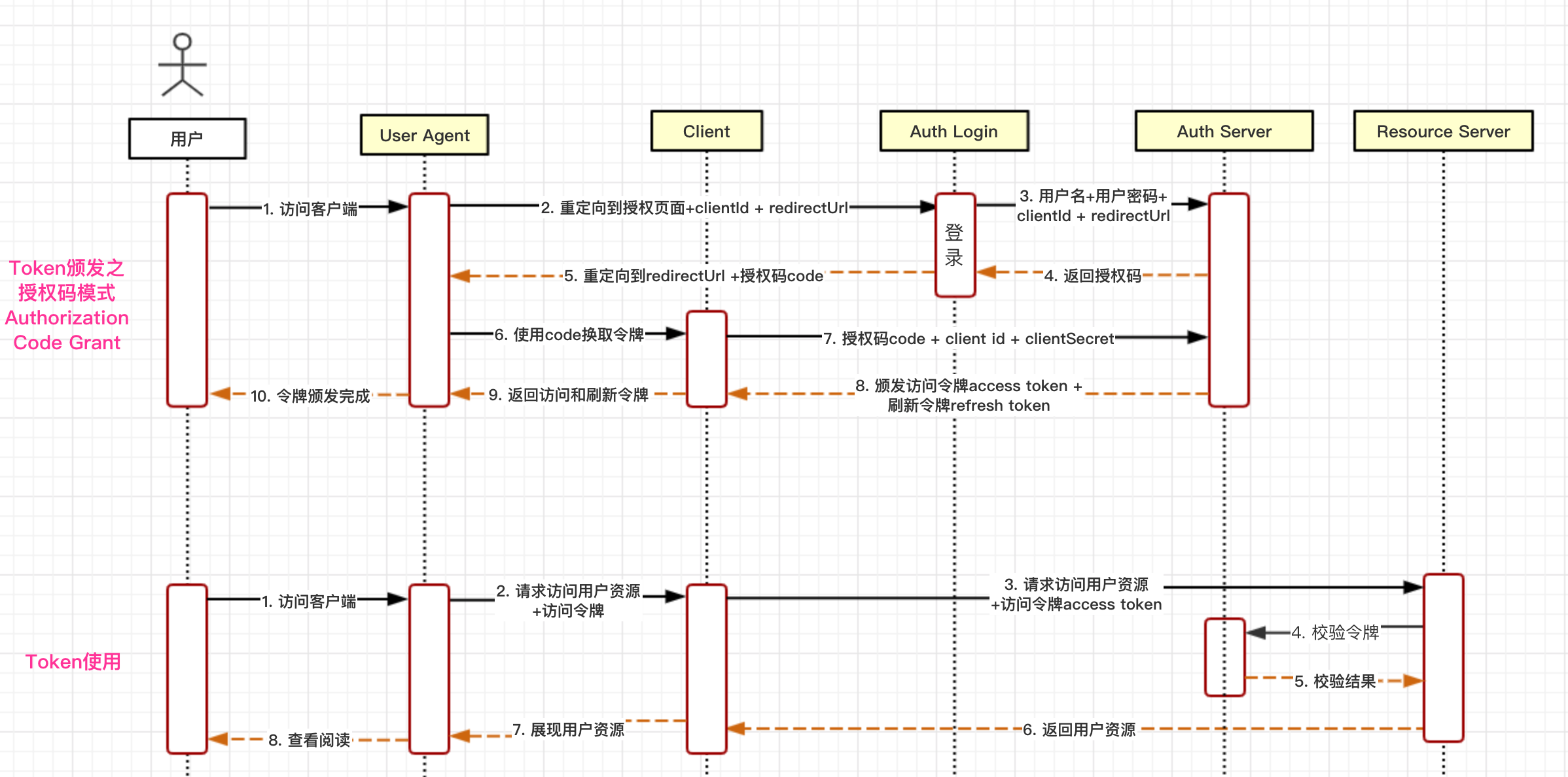
客户端必须得到用户的授权（authorization grant），才能获得令牌（access token）。OAuth 2.0定义了四种授权方式。

* 授权码模式（authorization code）
* 简化模式（implicit）
* 用户授信模式（resource owner password credentials）
* 应用授信模式（client credentials）

### 授权码模式-Authorization Code Grant：

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。

####授权码流程图：



关键步骤：

获取用户的授权码：

# 用户的认证请求： Authorization Request  
  
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  
   
参数解读：  
请求方法：GET，  
- response\_type：表示授权类型，必填项，此处的值固定为"code"  
- client\_id：表示客户端的ID，必填项  
- 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。  
  
# 用户的认证响应：Authorization Response  
# 响应成功  
HTTP/1.1 302 Found  
Location: https://client.example.com/cb?code=SplxlOBeZQQYbYS6WxSbIA&state=xyz  
  
参数解读：  
- 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。  
  
# 响应失败：  
HTTP/1.1 302 Found  
Location: https://client.example.com/cb?error=access\_denied&state=xyz  
  
参数：  
error：返回的必选项，以下内容是值的可选项：  
 invalid\_request：请求参数错误；例如参数的个数的多少，包括错误的参数等。  
 unauthorized\_client：没有被认证的客户端，主要原因是否为官方注册的应用。  
 access\_denied:用户主动取消授权。  
 unsupported\_response\_type：在请求的时候传递的不支持的code字符。  
 invalid\_scope：非法的权限参数。  
 server\_error:服务器错误。  
 temporarily\_unavailable:临时的由于服务器的负载或者在维护期间导致时的不可用  
state： 客户端传递的state值。  
error\_description：可选项，错误信息的描述原因。  
error\_uri:可选项，用户提供给可读性比较高的展示错误的页面。

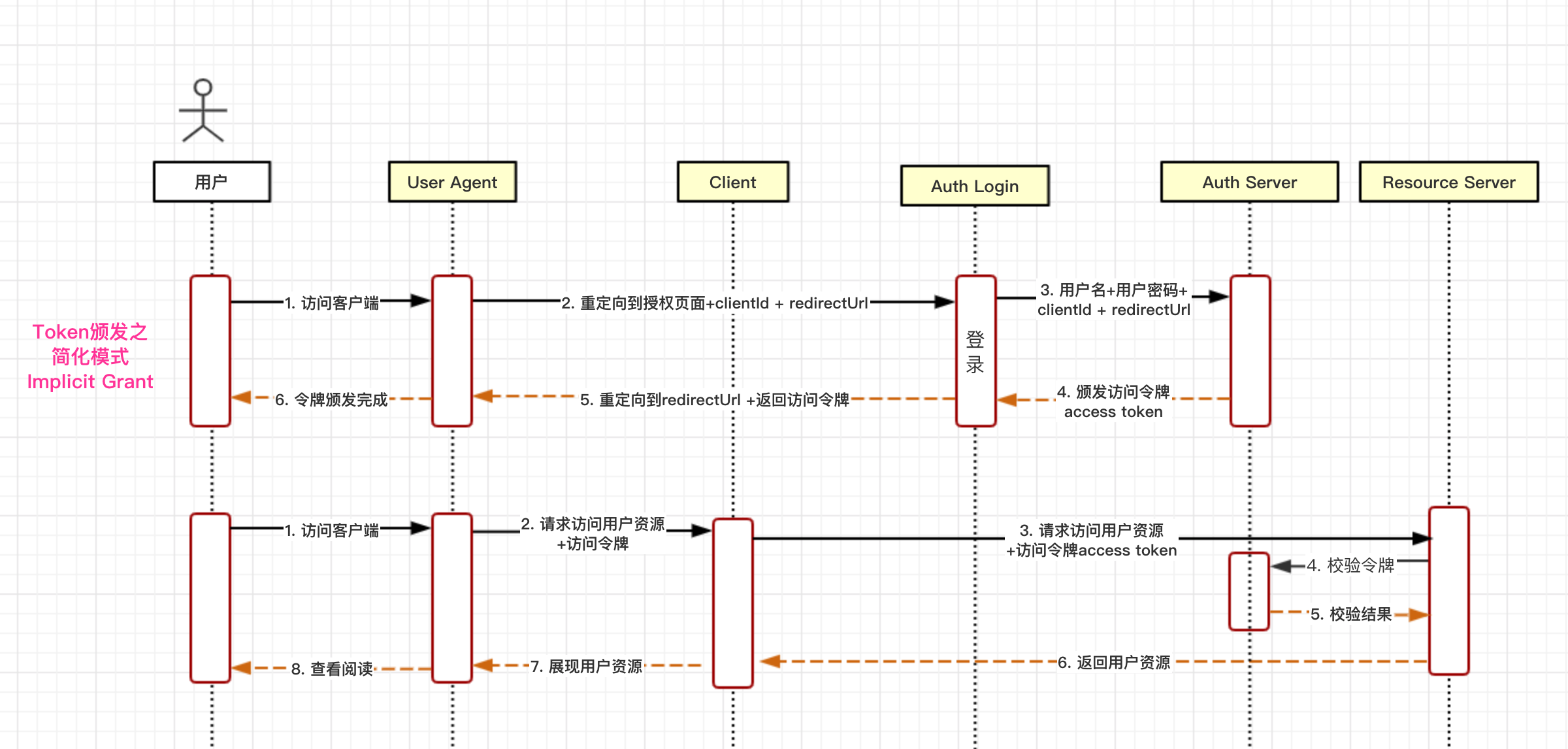
获取用户的access\_token,此步骤在需要在获取用户的授权码的基础上进行:

# Access Token Request  
# 请求示例：  
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&client\_id=12343543654643  
  
# 参数解读：  
- 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。  
  
# 响应示例：  
# Access Token Response  
  
  
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"  
}  
  
# 参数解读：  
 access\_token：授权令牌，可以根据用户的授权范围获取用户信息。  
 token\_type: 授权类型，无实际应用。  
 expires\_in: 有效时间，一般为秒。  
 refres\_token:在access\_token失效之后用来获取新token。不过被部分应用废弃，如果授权过期，则需要重新获取授权。  
 example\_parameter：无实际意义，可能还有的其他参数。

### 简化模式-Implicit Grant：

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。

#### 简化模式流程图：



关键步骤：

此过程通过获取用户授权页面之后，用户在授权页面授权，之前在前端的浏览器中重定向查询参数中返回access\_token.

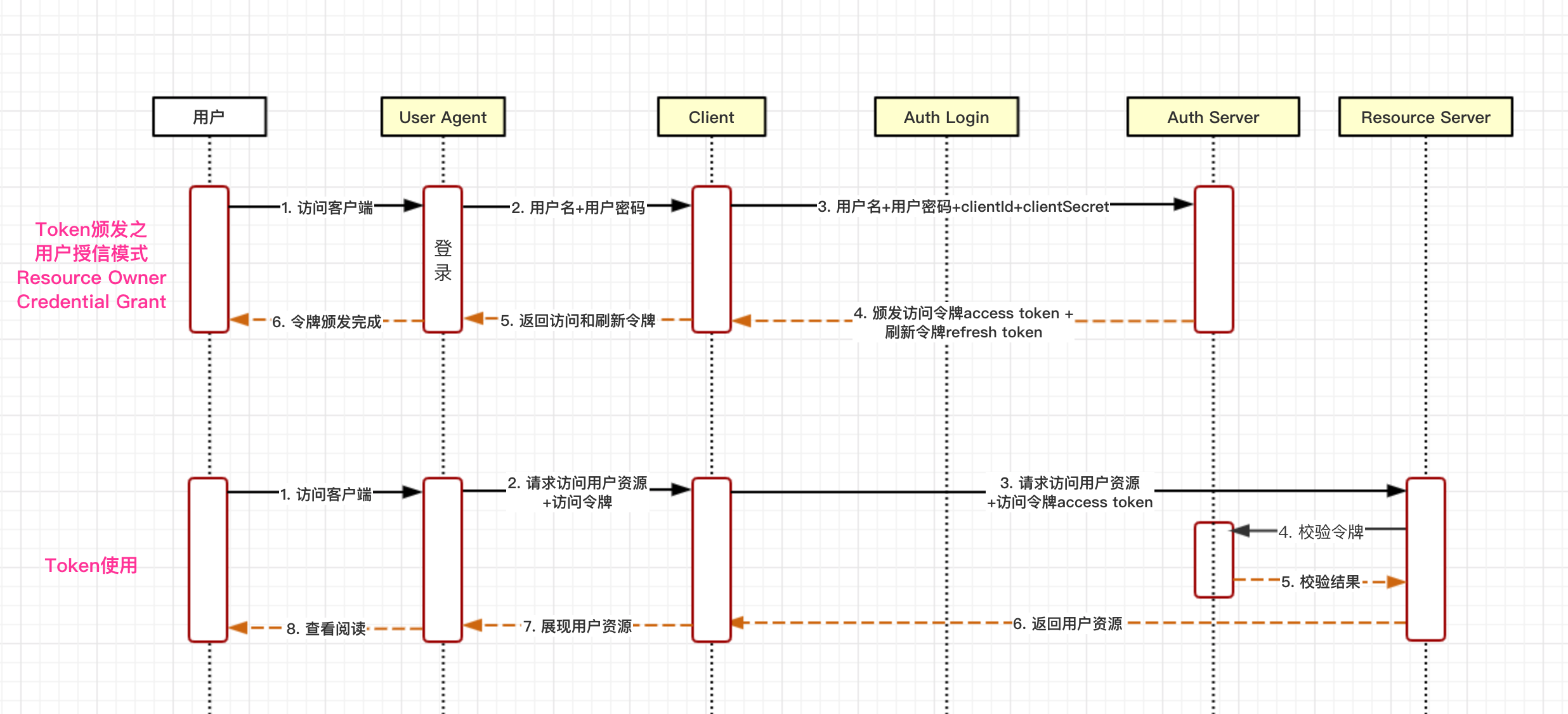
#请求示例：  
GET /authorize?response\_type=token&client\_id=s6BhdRkqt3&state=xyz  
 &redirect\_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1  
Host: server.example.com  
  
# 参数解读：  
- response\_type：表示授权类型，此处的值固定为"token"，必选项。  
- client\_id：表示客户端的ID，必选项。  
- redirect\_uri：表示重定向的URI，可选项。  
- scope：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。   
  
  
# 正常响应：Access Token Response  
# 响应示例  
HTTP/1.1 302 Found  
Location: http://example.com/cb#access\_token=2YotnFZFEjr1zCsicMWpAA  
 &state=xyz&token\_type=example&expires\_in=3600  
   
  
# 参数解读：  
- 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。  
  
# 异常响应：Error Response  
HTTP/1.1 302 Found  
Location: https://client.example.com/cb#error=access\_denied&state=xyz  
   
# 参数解读：  
参数：  
error：返回的必选项，以下内容是值的可选项：  
 invalid\_request：请求参数错误；例如参数的个数的多少，包括错误的参数等。  
 unauthorized\_client：没有被认证的客户端，主要原因是否为官方注册的应用。  
 access\_denied:用户主动取消授权。  
 unsupported\_response\_type：在请求的时候传递的不支持的code字符。  
 invalid\_scope：非法的权限参数。  
 server\_error:服务器错误。  
 temporarily\_unavailable:临时的由于服务器的负载或者在维护期间导致时的不可用  
state： 客户端传递的state值。  
error\_description：可选项，错误信息的描述原因。  
error\_uri:可选项，用户提供给可读性比较高的展示错误的页面。

### 用户授信模式-Resource Owner Password Credentials Grant：

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

#### 用户授信模式流程图



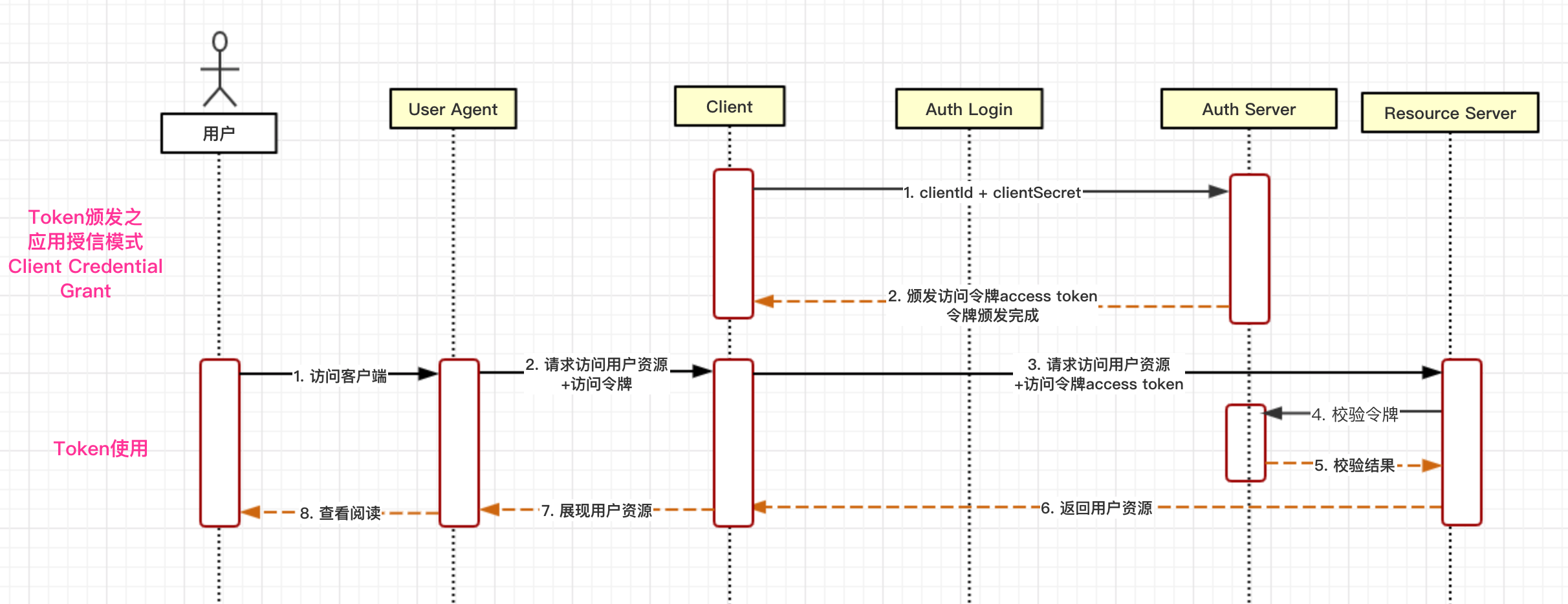
关键步骤：

# 请求示例：Access Token Request  
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  
   
# 参数解读：  
- 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。  
  
# 响应示例：Access Token Response  
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"  
}  
# 参数解读：  
- access\_token：授权令牌，可以根据用户的授权范围获取用户信息。  
- token\_type: 授权类型，无实际应用。  
- expires\_in: 有效时间，一般为秒。  
- refres\_token:在access\_token失效之后用来获取新token。不过被部分应用废弃，如果授权过期，则需要重新获取授权。  
- example\_parameter：无实际意义，可能还有的其他参数。

### 应用授信模式-Client Credentials Grant：

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。

#### 应用授信模式流程图：



关键步骤：

# 请求示例：Access Token Request  
POST /token HTTP/1.1  
Host: server.example.com  
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW  
Content-Type: application/x-www-form-urlencoded  
  
grant\_type=client\_credentials  
  
# 参数解读：  
- grant\_type：表示授权类型，此处的值固定为"client\*credentials"，必选项。  
- scope：表示权限范围，可选项。  
  
# 响应示例：Access Token Response  
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"  
 }  
# 参数解读：  
 access\_token：授权令牌，可以根据用户的授权范围获取用户信息。  
 token\_type: 授权类型，无实际应用。  
 expires\_in: 有效时间，一般为秒。  
 example\_parameter：无实际意义，可能还有的其他参数。