FRC - 参考指南

Jcohy

2021-09-13

Table of Contents

Chapter 1. rfc2617 - HTTP Authentication: Basic and Digest Access Authentication

原文链接：https://tools.ietf.org/html/rfc2617

Chapter 2. rfc4627 - The application/json Media Type for JavaScript Object Notation (JSON)

原文链接：https://tools.ietf.org/html/rfc4627

Chapter 3. rfc6749 - OAuth 2.0 授权框架

原文链接：https://tools.ietf.org/html/rfc6749

摘要

OAuth 2.0
授权框架允许第三方应用程序通过如下任意一种方式获取有限制的访问：

- 第三方应用代表资源拥有者发起在资源拥有者和HTTP服务之间的互动.
- 第三方应用通过其身份来获取访问权限.

本文取代并淘汰了在 RFC 5849 中所描述的的 OAuth 1.0 协议.

本备忘录的状态：这是 Internet 标准跟踪文档.

本文档是 Internet 工程任务组(IETF)的产品.它代表了 IETF 社区的共识.它获得了公众审查,并已获得互联网工程指导小组(IESG)批准发布.有关 Internet 标准的更多信息,请参见 RFC 5741 的第 2 节.
有关本文档当前状态,任何勘误以及如何提供反馈的信息,请访问 www.rfc-editor.org/info/rfc6749.

3.1. 介绍

在传统的 C/S 身份验证模型中
,客户端通过使用资源所有者的凭据向服务器进行身份验证来请求服务器上的访问受限资源(受保护资源). 为了向第三方应用程序能够访问受限资源,资源所有者需要与第三方共享其凭据.
这会产生一些问题和局限:

- 为了将来的需要,第三方程序需要存储资源拥有者的凭据(通常为明文密码)
- 即使密码验证存在安全漏洞,服务器仍然需要支持它
- 第三方程序对资源拥有者的受保护资源拥有过于宽泛的权限
  ,同时资源拥有者也没有能力对第三方程序进行限制(如限制第三方程序仅访问部分资源,或限制第三方程序的访问时间等)
- 资源拥有者必须通过更改密码来撤销第三方应用的权限.并且不能对单个第三方应用撤权
  (一旦更改密码,所有之前授予权限的第三方应用程序都要重新授权)
- 任意第三方应用的泄密都会导致终端用户的密码和受该密码保护的所有数据泄密.

OAuth 通过引入授权层并将客户端的角色与资源所有者的角色分开来解决这些问题.在 OAuth 中,客户端请求访问由资源所有者拥有并由资源服务器托管的资源,并向其颁发与资源所有者不同的凭据集.

客户端通过获取访问令牌而不是使用资源拥有者的凭据来访问受限资源.访问令牌是一个表示特定作用域,生命周期和其他访问属性的字符串.授权服务器在资源所有者的批准下才会向第三方客户端颁发访问令牌.客户端使用访问令牌来访问资源服务器托管的受保护资源.

例如,一个终端用户(资源拥有者)可以授权打印服务(客户端)访问存储在照片共享服务(资源服务器)中的受保护照片,而无需与打印服务共享其用户名和密码.相反,她直接使用照片共享服务信任的服务器(授权服务器)进行身份验证,该服务器向打印服务发放特定凭证(访问令牌).

此规范旨在与 HTTP (RFC2616) 一起使用.在 HTTP 之外的任何协议上使用 OAuth 都超出了本规范的范围.

作为信息文档发布的 OAuth 1.0 协议(RFC5849)是一个小型临时社区工作的结果.此标准跟踪规范建立在 OAuth 1.0 部署经验的基础上,以及从更广泛的 IETF 社区收集的其他用例和可扩展性要求.OAuth 2.0 协议与 OAuth 1.0 不向后兼容.这两个版本可以在网络上共存,并且实现可以选择支持两者.但是,本规范的目的是所有的新系统均采用 OAuth2.0,OAuth1.0 仅用于支持已经部署的系统.OAuth 2.0 协议与 OAuth 1.0 协议共享的实现细节很少,所以熟悉 OAuth 1.0 的实施者不应该对本规范的结构和细节进行臆测.

3.1.1. 角色

OAuth定义了四个角色:

资源所有者(resource owner)

能够对受保护资源授予访问权限的实体.当资源所有者是一个人时,它被称为终端用户.

资源服务器(resource server)

托管受保护资源的服务器,能够接受和响应通过令牌对受保护的资源的请求.

客户端(client)

代表资源所有者及其授权进行受保护资源请求的应用程序.术语"客户端"并不暗示任何特定的实现特征(例如,应用程序是在服务器,台式机还是其他设备上执行).

授权服务器(authorization server)

成功后,服务器向客户端发出访问令牌验证资源所有者并获得授权.

授权服务器和资源服务器之间的交互超出了本规范的范围.授权服务器可以是与资源服务器相同的服务器或单独的实体.单个授权服务器可以发出可以被多个资源服务器接受的访问令牌.

3.1.2. 协议流程

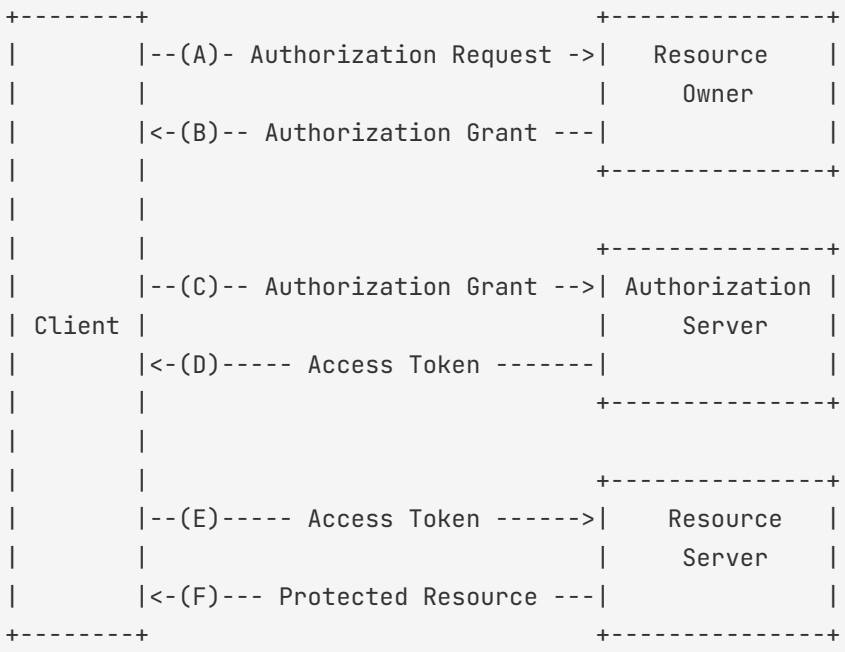


图 1: Abstract Protocol Flow

Figure 1中所示的抽象 OAuth 2.0 流程描述了四个角色之间的交互.包括以下步骤:

A. 客户端请求资源所有者授权.该授权请求可以直接直接呈现给资源拥有者,也可间接地通过授权服务器进行(例如跳转到授权服务器).

B. 客户端收到授权许可,即表示资源所有者授权的凭证,使用本规范中定义的四种授权类型之一或使用扩展授权类型表示.授权许可类型取决于客户端使用何种方法请求授权服务器,以及授权服务器支持哪些授权类型.

C. 客户端通过向授权服务器进行认证并呈现用户赋予的权限来请求 access token.

D. 授权服务器验证客户端并验证用户赋予的权限,如果有效,则颁发 access token.

E. 客户端从资源服务器请求受保护资源,并通过呈现 access token 进行身份验证.

F. 资源服务器验证 access token,如果有效,则为该请求提供服务.

客户端从资源所有者获得授权授权的首选方法(如步骤(A)和(B)所示)是使用授权服务器作为中介

,如第 4.1 节中的图 3所示.

3.1.3. 权限授予

权限授予(Authorization Grant)是资源拥有者同意授权请求(访问受保护资源)的凭据,客户端可以用它来获取 access token. 本规范定义了四种授权(grant)类型 - 授权码模式(authorization code),简化模式(implicit),密码模式(resource owner password credentials)和客户端模式(client credentials),以及用于定义其他类型的可扩展性机制.

授权码模式(Authorization Code)

授权码是通过授权服务器来获得的,授权服务器是客户端和资源拥有者之间的媒介.与客户端直接向资源拥有者申请权限不同,客户端通过将资源拥有者引向授权服务器(通过 RFC 2616 中定义的 user-agent),然后授权服务器反过来将资源拥有者 redirect 到 client(附带上 authorization code).

在将资源拥有者 redirect 到 client(附带 authorization code)之前,授权服务器验证资源拥有者并获取授权.因为资源拥有者仅与授权服务器进行身份验证,所以资源拥有者的凭据(用户名、密码等)永远不会泄露给客户端(尤其是第三方客户端).

授权码有一些重要的安全优势,比如验证 client 的能力,比如直接将 access token 传送给 client 而不是通过资源拥有者的 user-agent(可能会将token泄露给第三方).

简化模式(implicit)

简化模式是为在浏览器中使用诸如 JavaScript 之类的脚本语言而优化的一种简化的授权码流程.在简化模式中,直接将 access token 而不是 authorization code 颁发给 client(通过资源拥有者的授权).grant 类型为 implicit,所以没有中间环节(比如用来在稍后获取 access token 的 authorization code)

在简化模式中颁发 access token 时,授权服务器没有对 client 进行验证.在某些情况下,可以通过用来获取 access token 的重定向 URI 来验证 client.access token 可以通过访问资源拥有者的 user-agent 暴露给资源拥有者或者其他的应用.

由于简化模式减少了获取 access token 的往返次数,所以可以提高某些客户端的响应能力和效率(比如一个运行在浏览器中的应用). 但是

,应该权衡使用简化模式所带来的便捷性与其带来的安全隐患之间的利害关系(在 10.3 和 10.16 中有描述),尤其是授权码模式可用时.

密码模式(resource owner password credentials)

资源拥有者密码凭据(如用户名和密码)可以用来直接用来当做一种获取 access token 的权限授予方式.凭据仅应当在资源拥有者高度信任 client 时使用(比如 ,应用是设备操作系统的一部分,或有较高权限的应用),并且其他授权模式(比如授权码模式)不可用时.

尽管这种授权类型需要 client 直接接触资源拥有者的凭据 ,资源拥有者的凭据仅被用于单次的获取 access token 的请求 .通过使用用户凭据来交换具有较长寿命的 access token 或者 refresh token,这种授权模式可消除 client 在将来需要授权时对资源拥有者凭据的需求(就是说 ,这次通过用户凭据获取了 access token,以后就可以直接通过 access token 而不是用户凭据来访问受限资源了).

客户端模式(client credentials)

当授权范围限于客户端控制下的受保护资源或先前与授权服务器一起安排的受保护资源时,client 凭据(或其他形式的客户端身份验证)可用作权限授予.客户端凭证通常是在客户端代表自己(客户端也是资源所有者)或基于先前与授权服务器一起安排的授权请求访问受保护资源时用作权限授予.

3.1.4. 访问令牌(Access Token)

access token是用来访问受限资源的凭据.access token 是一个代表授予 client 的权限的字符串.该字符串通常对 client 不透明.token 表示特定范围和持续时间的访问权限,由资源所有者授予,由资源服务器和授权服务器执行.

令牌可以表示用于检索授权信息的标识符,或者可以以可验证的方式自包含授权信息(即,由一些数据和签名组成的令牌串).client 可能需要额外的身份验证凭据(超出本规范的范围) 来使用令牌.

访问令牌提供一个使用单个的资源服务器可以理解的令牌来替换其他不同的身份验证方式(如用户名+密码方式)的抽象层.这种抽象使得颁发访问令牌比用于获取它们的权限授予更具限制性,并且消除了资源服务器理解各种不同身份验证方法的需要.

访问令牌可以具有基于资源服务器安全性要求的不同格式,结构和使用方法(例如,加密属性).访问令牌属性和用于访问受保护资源的方法超出了本规范的范围,并由协同规范(如 RFC6750)定义.

3.1.5. 刷新令牌(Refresh Token)

refresh token 是用于获取 access token 的凭据.refresh token 由授权服务器颁发给 client,用于在当前访问令牌变为无效或过期时获取新的访问令牌,或者获取具有相同或更窄范围的其他访问令牌(访问令牌可能具有更短的生命周期和权限少于资源所有者授权的权限.根据授权服务器的判断,发出刷新令牌是可选的.如果授权服务器发出刷新令牌,则在发出访问令牌时包括它(即图1中的步骤(D)).

刷新令牌是表示资源所有者授予客户端的权限的字符串.该字符串通常对客户端不透明.令牌表示用于检索 授权信息的标识符.与访问令牌不同,刷新令牌仅用于授权服务器
,不会发送到资源服务器.

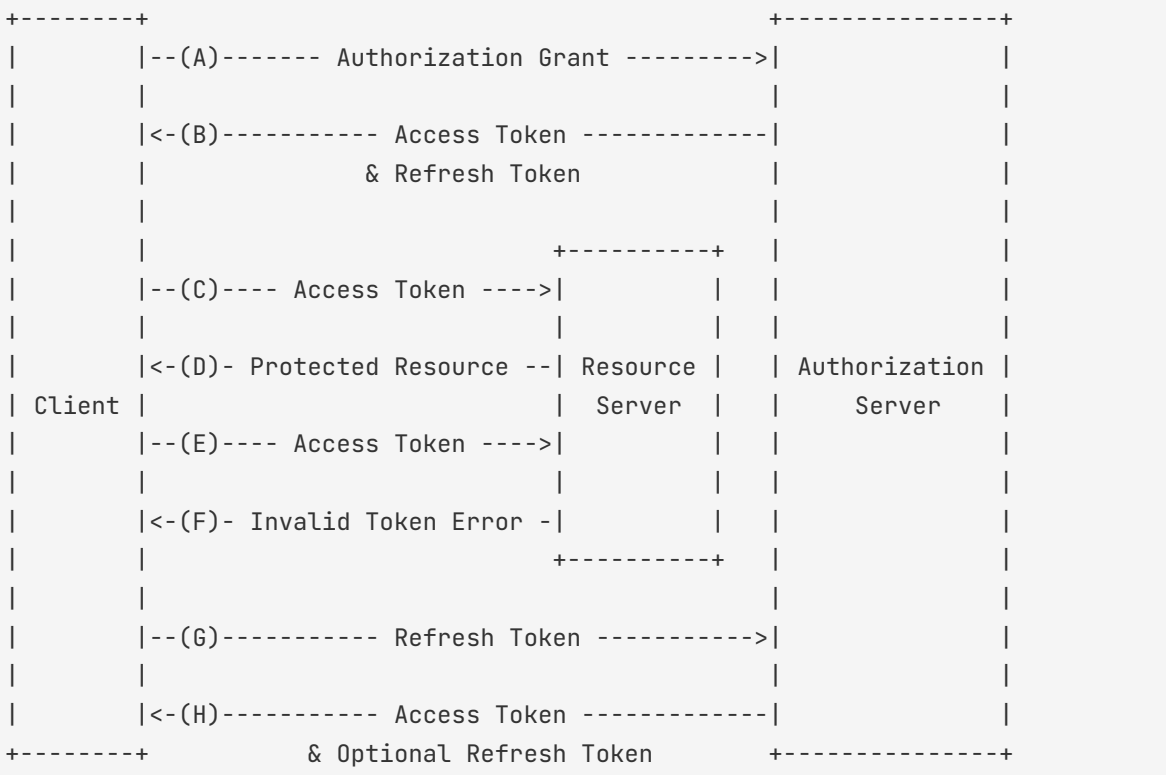


图 2: Refreshing an Expired Access Token

图2所示的流程包括以下步骤:

A. 客户端通过向授权服务器进行认证、发起权限授予来获取 access token.

B. 授权服务器验证客户端并验证权限授予授权,如果有效,则颁发访问令牌和刷新令牌.

C. 客户端通过呈现访问令牌向资源服务器发出受保护的资源请求.

D. 资源服务器验证访问令牌,如果有效,则为请求提供服务.

E. 重复步骤©和(D)直到访问令牌到期.如果客户端知道访问令牌已过期,则跳到步骤(G);否则,它会生成另一个受保护的资源请求

F. 由于访问令牌无效,资源服务器返回无效的令牌错误.

G. 客户端通过向授权服务器进行身份验证并显示刷新令牌来请求新的访问令牌.该客户端身份验证的要求是基于客户端类型和授权服务器策略.

H. 授权服务器验证客户端并验证刷新令牌,如果有效,则发出新的访问令牌(以及可选的新刷新令牌).

步骤©,(D),(E)和(F)不属于规范的范围,如 第 7 节所述.

3.1.6. TLS 版本

由于广泛的部署和已知的安全性漏洞,当本规范使用安全传输层协议(TLS)时可能存在不同的适用版本.在本协议发表时,TLS v1.2 RFC5246 是最新版本,但是部署基础非常有限,可能无法实现.TLS v1.0 RFC2246是最广泛的部署版本并将提供最广泛的互操作性.实现还可以支持满足其安全要求的其他传输层安全机制.

3.1.7. HTTP 重定向(HTTP Redirections)

在 client 话说授权服务器将 user-agent 导向另一个目的地时,本规范广泛地使用了 HTTP 重定向.虽然本规范中的示例使用 HTTP 302 状态代码进行重定向,但是允许其他的实现通过其他方法实现重定向,这也被认为是实现细节的一部分.

3.1.8. 互通性

OAuth 2.0 提供了一个具有明确定义的具有丰富的安全属性的授权框架.但是,作为一个具有许多可选组件的丰富且高度可扩展的框架,该规范本身可能会产生各种不可互操作的实现.

此外,对于一些组件,本规范仅有部分定义或完全未定义(例如,客户端注册,授权服务器功能,end

point 发现).如果没有这些组件
,客户端必须专门手动地针对特定授权服务器和资源服务器进行配置以进行互操作.

该框架的设计明确期望未来的工作将定义实现完整的Web级互操作性所必需的规范性配置文件和扩展.

3.1.9. 符号约定(Notational Conventions)

本规范中的关键字 "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", 和 "OPTIONAL" 应按 RFC2119 中所述进行解释.

3.2. 客户端注册

在启动协议之前,client 向授权服务器注册.client
注册的方式使用授权服务器超出了本规范的范围,但通常涉及终端用户与 HTML 注册表单的交互.

客户端注册不需要客户端和授权服务器之间的直接交互.当授权服务器支持时,注册可以依赖于其他方式来建立信任并获得所需的客户端属性(例如,重定向 URI,客户端类型).例如
,可以使用自发布或第三方发布的断言来完成注册,或者通过使用
可信通道执行客户端发现的授权服务器来完成注册.

注册客户端时,客户端开发人员应该:

- 指定 如第 2.1 节所述的客户端类型
- 提供如 第 3.1.2 节所述的 client 重定向 URI, 以及
- 包含授权服务器所需的任何其他信息(例如,应用程序名称,网站,描述,徽标图像
  ,所接受的法律条款).

3.2.1. 客户端类型(Client Types)

OAuth 根据其与授权服务器进行安全身份验证的能力定义了两种客户端类型(即
,保证其客户凭证的机密性的能力):

机密

客户端能够维护其凭证的机密性(例如
,在具有对客户端凭证具有受限访问的安全服务器上实现的客户端),或能够使用其他方式进行安全的客户端认证.

公开

客户端无法维护其凭据的机密性(例如,在资源所有者使用的设备上执行的客户端
,例如已安装的本地应用程序或基于Web 浏览器的应用程序
),并且无法通过任何其他方式进行安全的客户端身份验证.

客户端类型标识基于授权服务器的安全身份验证定义及其可接受的客户端凭据暴露级别.授权服务器不应该对客户端类型做出假设.

客户端可以被实现为分布式组件集,每个组件具有不同的客户端类型和安全性上下文(例如,具有基于机密服务器的组件和基于公共浏览器的组件的分布式客户端).如果授权服务器不提供对此类客户端的支持或不提供有关其注册的指导,则客户端应该将每个组件注册为单独的客户端.

此规范是围绕以下客户端配置设计的:

Web应用程序

Web应用程序是在Web 服务器上运行的机密客户端
.资源所有者通过在资源所有者使用的设备上的用户代理中呈现的HTML用户界面来访问客户端.客户端凭据以及发布到客户端的任何访问令牌都存储在Web服务器上,不会向资源所有者公开或访问.

基于用户代理的应用程序

基于用户代理的应用程序是公共客户端,其中客户端代码从
web服务器下载并在资源所有者使用的设备上的用户代理(例如,web浏览器)内执行.协议数据和凭证可以轻松访问(并且通常可见)资源所有者.由于此类应用程序驻留在用户代理中,因此它们可以在请求授权时无缝使用用户代理功能.

本地应用程序

本地应用程序是在资源所有者使用的设备上安装和执行的公共客户端
.资源所有者可以访问协议数据和凭证.这是假设的可以提取应用程序中包含的任何客户端身份验证凭据.另一方面,动态发布的凭证(例如访问令牌或刷新令牌)可以获得可接受的保护级别.至少,这些凭据受到保护,从而免受应用程序可能与之交互的恶意服务器的影响.在某些平

台上,可能会保护这些凭据免受驻留在同一设备上的其他应用程序的影响.

3.2.2. 客户端标识(Client Identifier)

授权服务器向已注册的 client 颁发 client identifier - 一个代表该 client 注册信息的唯一字符串.client identifier 不需要保密 ,它被暴露给资源拥有者并且禁止单独用于 client 认证 .客户端标识符对于授权服务器是唯一的. 本规范未定义 client identifier 字符串的大小 .客户端应避免对标识符大小进行假设.授权服务器应该记录它发出的任何标识符的大小.

3.2.3. 客户端认证(Client Authentication)

如果客户端类型是机密的,则客户端和授权服务器建立适合授权服务器的安全性要求的客户端认证方法.授权服务器可以接受满足其安全要求的任何形式的客户端身份验证.

机密客户端通常被颁发(或建立)用于与授权服务器进行认证的一组客户机凭证(例如,密码,公钥/私钥对).

授权服务器可以与公共客户端建立客户端身份验证方法.但是,授权服务器不得依赖公共客户端身份验证来识别客户端.

客户端在每个请求中最多使用一种身份验证方法.

客户端密码(Client Password)

拥有客户端密码的客户端可以使用 RFC2617 中定义的 HTTP Basic 身份验证方案向授权服务器进行身份验证.使用 附录B:使用 application/x-www-form-urlencoded 媒体类型 编码算法对客户端标识符进行编码, 并将编码 value 用作 username;客户端密码使用相同的算法进行编码并用作 password.授权服务器必须支持 HTTP基本身份验证方案,以便对发出客户端密码的客户端进行身份验证.

例如(额外换行符仅用于排版目的):

```
Authorization: Basic czZCaGRSa3F0Mzo3RmpmcDBaQnIxS3REUmJuZlZkbUl3
```

或者,授权服务器可以选择支持在请求体中包含如下参数的客户端凭据:

client_id

REQUIRED.在 2.2 节描述的注册过程中发给客户端的客户端标识符.

client_secret

REQUIRED. The client secret. 如果客户端密钥是空字符串,
则客户端可以省略该参数.

使用这两个参数在请求体中包含客户端凭证是不推荐的,并且应该仅限于无法直接使用 HTTP 基本身份验证方案(或其他基于密码的 HTTP 身份验证方案)的客户端.
参数只能在请求体中传输,绝不能包含在请求 URI 中.

例如,使用 body 参数刷新访问令牌(第 6 节)的 HTTP 请求(额外换行符仅用于排版目的):

```
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
```

当使用密码验证发送请求时,授权服务器必须要求使用 如 1.6 节所述的TLS.

由于此客户端身份验证方法涉及密码,因此授权服务器必须保护使用它的任何 endpoint 免受穷举攻击.

其他认证方法

授权服务器可以支持符合其安全要求的任何合适的HTTP认证方案.使用其他身份验证方法时,授权服务器必须定义客户端标识符(注册记录)和身份验证方案之间的映射.

3.2.4. 未注册的客户端(Unregistered Clients)

此规范不排除使用未注册的客户端.但是,此类客户端的使用超出了本规范的范围,需要进行额外的安全性分析并检查其互操作性影响.

3.3. 协议端点(Protocol Endpoints)

授权过程使用两个授权服务器端点(HTTP资源):

Authorization endpoint

客户端使用该端点通过用户代理重定向从资源所有者获取授权.

Token endpoint

客户端用于通过 user-agent redirection 从资源所有者获取授权.

以及一个客户端端点:

Redirection endpoint

授权服务器用于通过资源所有者 user-agent将包含授权凭据的响应返回给客户端.

并非每种授权授权类型都使用两个端点.扩展授权类型可以根据需要定义其他端点.

3.3.1. 授权端点(Authorization endpoint)

授权端点用于与资源所有者交互并获得权限授予. 授权服务器必须首先验证资源所有者的身份.
授权服务器验证资源所有者的方式(例如,用户名和密码登录,会话 cookie)
超出了本规范的范围.

客户端获取授权端点位置的方法超出了本规范的范围,因为这个位置通常由服务文档提供.

端点 URI 可以包括 application/x-www-form-urlencoded 格式(根据 附录B)的查询组件(RFC3986第 3.4 节),并且在添加其他查询参数时该组件必须保留.端点 URI 绝不能包含片段组件.

由于对授权端点的请求导致用户身份验证和凭据的明文传输(在HTTP响应中),在向授权端点发送请求时,授权服务器必须使用 第 1.6 节中所述的 TLS.对于没有值的参数 ,必须当作在请求中省略了该参数.授权服务器必须忽略无法识别的请求参数.请求和响应参数不得被包含多次.

授权服务器必须支持对授权端点使用 HTTP "GET" 方法 RFC2616,并且也可以支持使用 "POST" 方法.

如果请求参数没有携带任何值,则必须忽略.授权服务器必须忽略 无法识别的请求参数 .请求和响应参数 不得超过一次.

响应类型

授权终端由授权代码模式和简化授权模式的工作流中使用.客户端使用以下参数通知授权服务器所需的授权类型:

response_type

REQUIRED. 值必须是用于请求如 如第 4.1.1 节所述授权码的 "code" 或者 如 4.2.1 节所述用于请求访问令牌(简化授权)的 "token" ,或者注册的扩展值,如 第 8.4 节.

扩展响应类型可以包含空格 (%x20) 分隔的值列表,其中值的顺序无关紧要(例如,响应类型 "a b" 与 "b a" 相同).这种复合响应类型的含义由它们各自的规范定义.

如果授权请求缺少"response_type”参数,或者不理解响应类型,授权服务器必须返回 如第 4.1.2.1 节所述的错误响应.

重定向端点(Redirection Endpoint)

完成与资源所有者的交互后,授权服务器将资源所有者的用户代理指向客户端.在用户注册过程中或在发出授权请求时,授权服务器将 user-agent
重定向到先前与授权服务器建立的客户端重定向端点. 重定向端点URI必须是 RFC3986第 4.3 节 定义的绝对 URI.端点URI可以包括 application/x-www-form-urlencoded 格式的(附录B)查询组件(RFC3986第 3.4 节),并且在添加其他查询参数时必须保留该组件.端点 URI 绝不能包含片段组件.

加密请求端点(Endpoint Request Confidentiality)

当请求的响应类型是 code 或 token 时
,或者当重定向请求将导致在开放网络上传输敏感凭证时,重定向端点应该使用如 第 1.6 节所述的TLS.此规范并未强制要求使用 TLS,因为在撰写本文时,要求客户端部署 TLS对许多客户端开发人员来说是一个重大障碍.如果TLS不可用,授权服务器应该在重定向之前警告资源所有者关于不安全端点(例如,在授权请求期间显示消息).

缺乏TLS会严重影响客户端及其授权访问的受保护资源的安全性.当授权过程以委托的终端用户授权的形式被用作客户端(例如,第三方登录服务)使用时,TLS的使用尤其重要.

注册要求(Registration Requirements)

授权服务器必须要求以下客户端注册其重定向端点:

- 公共客户端
- 使用简化授权类型的机密客户端.

授权服务器应该在使用授权端点之前要求所有客户端注册其重定向端点.

授权服务器应该要求客户端提供完整的重定向 URI(客户端可以使用 `state` 请求参数来实现按请求定制)如果要求注册完整的重定向URI是不可能的,授权服务器应该要求注册URI方案,权限和路径(允许客户端在请求授权时仅动态改变重定向URI的查询组件).

授权服务器可以允许客户端注册多个重定向端

若无重定向注册,攻击者可能使用授权端点用作开放重定向器(如第 10.15 节中所述).

动态配置(Dynamic Configuration)

如果已注册了多个重定向 URI,或者只注册了部分重定向 URI,或者没有注册重定向 URI,则客户端必须使用 `redirect_uri` 请求参数包含带有授权请求的重定向 URI.

当授权请求中包含重定向 URI 时,如果注册了任一重定向 URI,授权服务器必须将接收到的值与按照 RFC3986 第 6 节中定义 的重定向 URI列表中至少一个已注册重定向 URI(或 URI 组件)进行匹配 .如果客户端注册包含完整重定向 URI,则授权服务器必须使用 RFC3986 第 6.2.1 节中定义的简单字符串比较法来比较两个 URI.

无效的端点(Invalid Endpoint)

如果授权请求由于重定向 URI 的缺少,无效或不匹配的而未通过验证 ,则授权服务器应当通知资源所有者该错误,并且不得自动将用户代理重定向到无效的重定向 URI.

端点内容(Invalid Endpoint)

对客户端点的重定向请求通常会导致由 `user-agent` 处理的 HTML 文档响应 (就是通常会返回一个网页).如果 HTML 响应直接作为重定向请求的结果提供,则 HTML 文档中包含的任何脚本都将以完全访问重定向 URI 及其包含的凭据的方式执行.

客户端不应在重定向端点响应中包含任何第三方脚本(例如,第三方分析,社交插件,广告网络).相反,它应该从URI中提取凭据并将用户代理再次重定向到另一个端点,而不暴露凭证(在 URI 或其他地方).如果包含第三方脚本,客户端必须确保首先执行自己的脚本(用于从 URI 中提取和删除凭据)

3.3.2. Token 端点(Token Endpoint)

客户端使用令牌端点通过呈现其权限授予或刷新令牌来获取访问令牌.除了简化授权类型之外,令牌端点与每个权限授予一起使用(因为访问令牌被直接发出).

客户端获取令牌端点位置的方法超出了本规范的范围,因为这通常由服务文档提供.

端点URI可以包括 `application/x-www-form-urlencoded` 格式的(附录B)查询组件(RFC3986 第 3.4 节),并且在添加其他查询参数时必须保留该组件.端点 URI 绝不能包含片段组件.

由于对 `token` 端点的请求导致用户身份验证和凭据的明文传输(在HTTP响应中),在向授权端点发送请求时,授权服务器必须使用 第 1.6 节中所述的TLS.对于没有值的参数,必须当作在请求中省略了该参数.授权服务器必须忽略无法识别的请求参数.请求和响应参数不得被包含多次.

客户端认证

在向令牌端点发出请求时,Confidential clients 或其他 clients 发出的客户端凭证必须使用授权服务器进行身份验证,如第 2.3 节所述.客户端身份验证用于:

- 强制将刷新令牌和授权码绑定到发给它们的客户端.当授权代码通过不安全的通道传输到重定向端点或者重定向 URI 尚未完整注册时,客户端身份验证至关重要.
- 通过禁用客户端或更改其凭据从受感染的客户端恢复,从而防止攻击者滥用被盗的刷新令牌.更改单组客户端凭据比撤消整组刷新令牌要快得多.
- 实施身份验证管理最佳实践,这需要定期进行凭据轮换.旋转整组刷新令牌可能具有挑战性,而单组客户端凭证的轮换则更加容易.

客户端可以在向令牌端点发送请求时使用 `client_id` 请求参数来对自身进行标识.在对 `token` 令牌端口的 `authorization_code grant_type` 请求中,未经身份验证的客户端必须发送其

client_id 以防止自己无意中接受用于具有不同 client_id 的客户端的代码.这可以保护客户端不会替换身份验证代码.(它不为受保护资源提供额外的安全性.)

3.3.3. 访问令牌范围

授权和令牌端点允许客户端使用 scope 请求参数指定访问请求的范围.反过来,授权服务器使用 scope 响应参数来通知客户端发出的访问令牌的范围.

scope 参数的值表示为以空格分隔的区分大小写的字符串列表.字符串由授权服务器定义.如果该值包含多个以空格分隔的字符串,则它们的顺序无关紧要,并且每个字符串都会为请求的范围添加其他访问范围.

```
scope       = scope-token *( SP scope-token )
scope-token = 1*( %x21 / %x23-5B / %x5D-7E )
```

根据授权服务器策略或资源所有者的指示,授权服务器可以完全或部分忽略客户端请求的范围.如果发布的访问令牌范围与客户端请求的范围不同,则授权服务器必须包含 scope 响应参数,以通知客户端授予的实际范围.

如果客户端在请求授权时省略了 scope 参数,则授权服务器必须使用预定义的默认值处理请求,或者使请求失败,指示范围无效.授权服务器应该记录其范围要求和默认值(如果已定义).

3.4. 获取授权

要请求访问令牌,客户端需从资源所有者处获取授权.授权以 authorization grant 的形式表示,客户端使用该 authorization grant 来请求访问令牌.OAuth 定义了四种授权类型 authorization code, implicit, resource owner password credentials, 和 client credentials.
它还提供了一种用于定义其他授权类型的扩展机制.

3.4.1. 授权码模式(Authorization Code Grant)

authorization code grant type 通常用于获取 access tokens 和 refresh tokens,并且针对 confidential clients 进行了优化. 由于这是一个基于重定向的流程,客户端必须有能力和资源拥有者的 user-agent(通常为WEB浏览器)进行交互,并且有能力接收来自授权服务器的重定向请求.

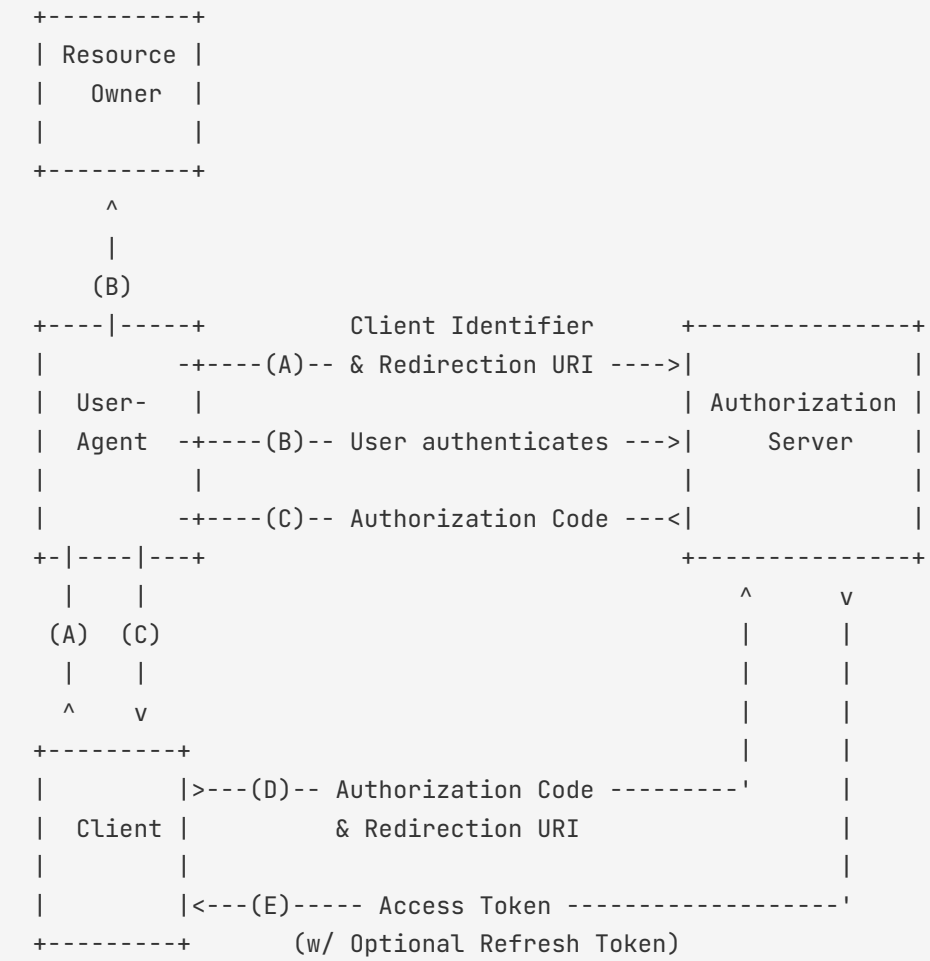


图 3: Authorization Code Flow

图 3中包含如下步骤:

授权请求(Authorization Request)

授权服务器对客户端进行身份验证,验证授权代码,并确保收到的重定向 *URI* 与步骤©中用于重定向客户端的*URI*相匹配.

客户端通过使用 `application/x-www-form-urlencoded` 格式将以下参数添加到授权端点 URI 的查询组件来构造请求URI(附录B):

response_type

必须的. 值必须为 `code`.

client_id

必须的. 客户端标识符,如第 2.2 节所述.

redirect_uri

可选的. 如第 3.1.2 节所述

scope

可选的. 访问请求的范围,如第 3.3 节所述.

state

推荐的. 客户端用于维护请求和回调之间状态的不透明值
.当授权服务器在将用户代理重定向回客户端时包含此值.这个参数应当被用来方式 CSRF 攻击 (参见 10.12 节).

客户端使用 HTTP 重定向响应或通过用户代理可用的其他方式将资源所有者定向到构造的URI.

例如,client 使用 TLS 将 user-agent 定向到如下 HTTP 请求:

```
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
```

为确保所有必须的参数都被呈现并且合法,授权服务器需要验证请求.如果请求合法,授权服务器对资源所有者进行身份验证并获取授权决策(通过询问资源所有者或通过其他方式建立批准).

建立决策时,授权服务器使用HTTP重定向响应或通过用户代理可用的其他方式将用户代理指向提供的客户端重定向 URI.

授权响应(Authorization Response)

如果资源所有者授予访问请求,则授权服务器通过使用 application/x-www-form-urlencoded 格式将以下参数添加到重定向 URI
的查询组件来发布授权代码并将其传递给客户端(参见附录B):

code

必须的. 授权服务器生成的授权码.授权代码必须在发布后尽快过期,以减少泄漏风险.建议最长授权代码生存期为 10 分钟.客户端不得多次使用授权码.如果授权代码被多次使用,授权服务器必须拒绝该请求,并且应该(如果可能)撤销先前基于该授权代码发出的所有令牌.授权代码与客户端标识符和重定向 URI 绑定.

state

必须的 如果客户端请求中存在 state 参数.则为从客户端接收到的值.

例如,授权服务器通过发送如下 HTTP 响应来重定向 user-agent:

```
HTTP/1.1 302 Found
Location: https://client.example.com/cb?code=SplxlOBeZQQYbYS6WxSbIA
          &state=xyz
```

客户端必须忽略无法识别的响应参数.本规范未定义授权代码字符串大小.客户端应避免对代码值大小进行假设.授权服务器应该记录由它发出的任何码值的大小.

错误响应(Authorization Response)

如果因为缺失、无效或不匹配的 URI,或者客户端的 identifier 缺失或无效而导致请求失败 authorization server 应当将这些错误通知给资源拥有者而不能自动将 user-agent 重定向到无效的 URI.

如果资源拥有者拒绝了访问请求,或因为其他原因失败,authorization server 应当以附录B所示的格式,以 application/x-www-form-urlencoded 的编码添加如下参数:

- error: 必需的 一个单个 ASCII USASCII 错误码.值域如下
  - invalid_request : 该请求缺少必需的参数,或者参数值无效,或者包含的参数超过一次,或者格式不正确.
  - unauthorized_client : 客户端未被授权,无法使用此方法
  - access_denied : 资源所有者或授权服务器拒绝了 请求.
  - unsupported_response_type : 授权服务器不支持使用此方法获取授权.
  - invalid_scope : 请求的范围无效,未知或格式错误.

- server_error : 授权服务器遇到意外情况导致其无法执行该请求.(此错误代码是必要的,因为500内部服务器错误HTTP状态代码不能由HTTP重定向返回给客户端).
- temporarily_unavailable : 授权服务器由于暂时超载或服务器维护目前无法处理请求.(此错误代码是必要的,因为503服务不可用HTTP状态代码不可以由HTTP重定向返回给客户端).

"error" 参数的值不能包含集合 %x20-21 / %x23-5B / %x5D-7E 以外的字符.

- error_description: 可选的. 提供错误描述的其他信息,用于协助客户端开发人员了解发生的错误. "error_description" 参数的值不能包含集合 %x20-21 / %x23-5B / %x5D-7E 的字符.
- error_uri:可选的. 提供一个错误页面,帮助客户端开发人员了解更多有关错误的其他信息. "error_uri" 参数的值必须符合 URI-reference 语法,因此不能包含集合 %x21 / %x23-5B / %x5D-7E 以外的字符.
- state:必须的 客户端授权请求中存在 state 参数 .则从客户端请求中获取.

例如,授权服务器通过发送以下HTTP响应重定向用户代理:

```
HTTP/1.1 302 Found
Location: https://client.example.com/cb?error=access_denied&state=xyz
```

访问令牌请求(Access Token Request)

客户端通过使用 附录B application/x-www-form-urlencoded 格式在 HTTP 请求实体正文中发送下列 UTF-8 字符编码的参数向令牌端点发起请求:

grant_type

必需的.值必须被设置为 authorization_code.

code

从授权服务器收到的授权码.

redirect_uri

必需的,若 redirect_uri 参数 如 4.1.1 节所述包含在授权请求中,且他们的值必须相同.

client_id

必需的,如果客户端没有和 如 3.2.1 节所述的授权服务器进行身份认证.

如果客户端类型是机密的或向客户端颁发了客户端凭据(或选定的其他身份验证要求),客户端必须按照第 3.2.1 节 所述与授权服务器进行身份验证.

例如,客户端使用 TLS 发起如下的 HTTP 请求(额外的换行符仅用于显示目的):

```
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
```

授权服务器必须: * 要求机密客户端或任何被颁发了客户端凭据(或有其他身份验证要求)的客户端进行客户端身份验证 * 如果包括客户端身份验证,则对客户端进行身份验证, * 确保授权码颁发给了通过身份验证的机密客户端,或者如果客户端是公开的,确保代码颁发给了请求中的 client_id, * 验证授权码是有效的,并 * 确保给出了 redirect_uri 参数,若 redirect_uri 参数 如 4.1.1 节 所述包含在初始授权请求中,确保它们的值是相同的.

访问令牌响应(Access Token Response)

如果访问令牌请求有效且已授权,则授权服务器发出访问令牌和可选的刷新令牌,如 第 5.1 节所述. 如果请求客户端身份验证失败或无效,授权服务器返回 如第 5.2 节所述的错误响应.

成功响应的例子

```
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"
}
```

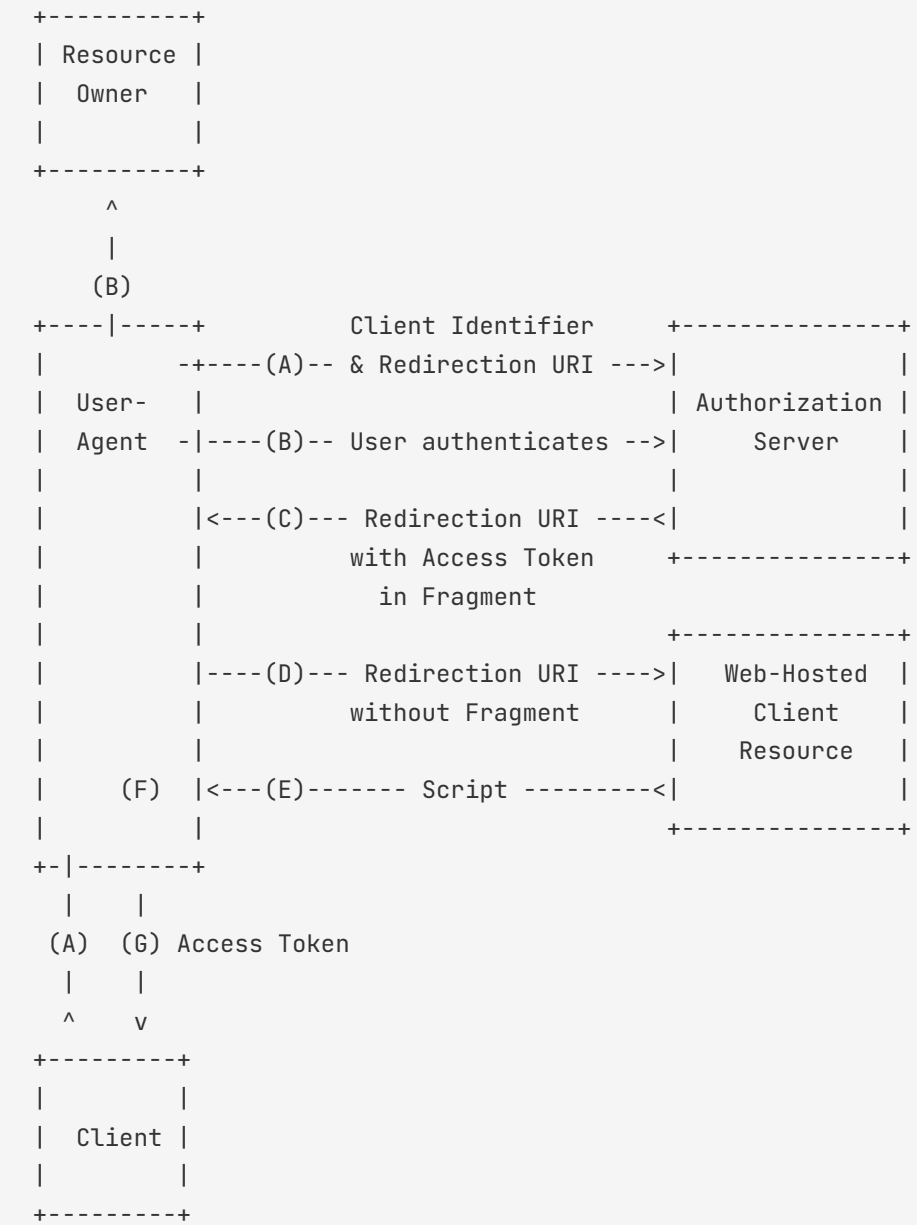
3.4.2. 简化模式(Implicit Grant)

简化授权类型被用于获取访问令牌(它不支持发行刷新令牌),并对知道操作具体重定向URI的公共客户端进行优化.这些客户端通常在浏览器中使用诸如 JavaScript 的脚本语言实现.

由于这是一个基于重定向的流程,客户端必须能够与资源所有者的用户代理(通常是 Web 浏览器)进行交互并能够接收来自授权服务器的传入请求(通过重定向).

不同于客户端分别请求授权和访问令牌的授权码模式类型,客户端收到访问令牌作为授权请求的结果.

简化许可类型(grant type)不包含客户端身份验证而依赖于资源所有者在场和重定向 URI 的注册.因为访问令牌被编码到重定向 URI 中
,它可能会暴露给资源所有者和其他驻留在相同设备上的应用.



Note: 说明步骤(A)和(B)的直线因为通过用户代理而被分为两部分.

图 4: Implicit Grant Flow

图4中的所示流程包含以下步骤:

A. 客户端通过向授权端点引导资源所有者的用户代理开始流程
.客户端包括它的客户端标识、请求范围、本地状态和重定向 URI,一旦访问被许可(
或拒绝)授权服务器将传送用户代理回到该 URI.

B. 授权服务器验证资源拥有者的身份(通过用户代理
),并确定资源所有者是否授予或拒绝客户端的访问请求.

C. 假设资源所有者许可访问,授权服务器使用之前(在请求时或客户端注册时)提供的重定向URI重定向用户代理回到客户端.重定向 URI 在 URI 片段中包含访问令牌.

D. 用户代理顺着重定向指示向 Web 托管的客户端资源发起请求(按 RFC2616 该请求不包含片段).用户代理在本地保留片段信息.

E. Web托管的客户端资源返回一个网页(通常是带有嵌入式脚本的 HTML 文档),该网页能够访问包含用户代理保留的片段的完整重定向 URI 并提取包含在片段中的访问令牌(和其他参数).

F. 用户代理在本地执行 Web 托管的客户端资源提供的提取访问令牌的脚本.

G. 用户代理传送访问令牌给客户端.

参见 第 1.3.2 节和 第 9 节 了解使用简化模式的背景.

参见 10.3 节和 10.16 节了解当使用简化模式时的重要安全注意事项.

授权请求(Authorization Request)

客户端通过按附录B使用"application/x-www-form-urlencoded”格式向授权端点URI的查询部分添加下列参数构造请求URI:

response_type

必需的.值必须设置为 `token`.

client_id

必需的.客户端标识符,如第 2.2 节所述.

redirect_uri

可选的.如第 3.1.2 节所述

scope

可选的.访问请求的范围,如第 3.3 节所述.

state

推荐的.客户端用于维护请求和回调之间状态的不透明值.当授权服务器在将用户代理重定向回客户端时包含此值.这个参数应当被用来方式 CSRF

攻击 (参见 10.12 节).

例如,客户端使用 TLS 定向用户代理发起下述 HTTP 请求(额外的换行仅用于显示目的):

```
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
```

授权服务器验证该请求,确保所有需要的参数已提交且有效.授权服务器必须验证它将重定向访问令牌的重定向URI与如 如第 3.1.2 节所述 的客户端注册的重定向URI匹配.

如果请求是有效的,授权服务器对资源所有者进行身份验证并获得授权决定(通过询问资源所有者或通过经由其他方式确定批准).

当确定决定后,授权服务器使用 HTTP 重定向响应向提供的客户端重定向 URI 定向用户代理,或者通过经由用户代理至该 URI 的其他可行方法.

访问令牌响应(Access Token Response)

如果资源所有者许可访问请求,授权服务器颁发访问令牌,通过使用按 附录B的 application/x-www-form-urlencoded 格式向重定向 URI 的片段部分添加下列参数传递访问令牌至客户端:

access_token

必需的.授权服务器颁发的访问令牌.

token_type

必需的.如 7.1 节所述的颁发的令牌的类型.值是大小写不敏感的.

expires_in

推荐的.以秒为单位的访问令牌生命周期.例如,值 3600 表示访问令牌将在从生成响应时的1小时后到期.如果省略,则授权服务器应该通过其他方式提供过期时间,或者记录默认值.

scope

可选的,若与客户端请求的范围相同; 否则,是必需的.如如第 3.3 节所述 的访问令牌的范围.

state

必须的 如果客户端请求中存在 state 参数.则为从客户端接收到的值.

授权服务器不能颁发刷新令牌.

例如,授权服务器通过发送以下 HTTP 响应重定向用户代理:(额外的换行符仅用于显示目的):

```
HTTP/1.1 302 Found
Location: http://example.com/cb#access_token=2YotnFZFEjr1zCsicMWpAA
          &state=xyz&token_type=example&expires_in=3600
```

开发人员应注意,一些用户代理不支持在 HTTP Location HTTP
响应头字段中包含片段组成部分.这些客户端需要使用除了 3xx
重定向响应以外的其他方法来重定向客户端——-例如,返回一个 HTML 页面
,其中包含一个具有链接到重定向 URI 的动作的"继续”按钮.

客户端必须忽略无法识别的响应参数.本规范未定义授权码字符串大小.客户端应该避免假设代码值的长度. 授权服务器应记录其发放的任何值的大小.

错误响应(Error Response)

如果因为缺失、无效或不匹配的 URI,或者客户端的 identifier 缺失或无效而导致请求失败 authorization server 应当将这些错误通知给资源拥有者而不能自动将 user-agent 重定向到无效的 URI.

如果资源拥有者拒绝了访问请求,或因为其他原因失败,authorization server 应当以附录B所示的格式,以 application/x-www-form-urlencoded 的编码添加如下参数:

- error: 必需的 一个单个 ASCII USASCII 错误码.值域如下
  - invalid_request : 该请求缺少必需的参数,或者参数值无效
    ,或者包含的参数超过一次,或者格式不正确.
  - unauthorized_client : 客户端未被授权使用此方法请求授权码.
  - access_denied : 资源所有者或授权服务器拒绝了请求.
  - unsupported_response_type : 授权服务器不支持使用此方法获取授权.

    - invalid_scope : 请求的范围无效,未知或格式错误.
    - server_error : 授权服务器遇到意外情况导致其无法执行该请求.(此错误代码是必要的,因为500内部服务器错误HTTP状态代码不能由HTTP重定向返回给客户端).
    - temporarily_unavailable : 授权服务器由于暂时超载或服务器维护目前无法处理请求.(此错误代码是必要的,因为503服务不可用HTTP状态代码不可以由HTTP重定向返回给客户端).

    "error" 参数的值不能包含集合 %x20-21 / %x23-5B / %x5D-7E 以外的字符.

- error_description: 可选的. 提供错误描述的其他信息,用于协助客户端开发人员了解发生的错误. "error_description" 参数的值不能包含集合 %x20-21 / %x23-5B / %x5D-7E 的字符.
- error_uri:可选的. 提供一个错误页面,帮助客户端开发人员了解更多有关错误的其他信息. "error_uri" 参数的值必须符合 URI-reference 语法,因此不能包含集合 %x21 / %x23-5B / %x5D-7E 以外的字符.
- state:必须的 客户端授权请求中存在 state 参数 .则从客户端请求中获取.

例如,授权服务器通过发送以下 HTTP 响应重定向用户代理:

```
HTTP/1.1 302 Found
Location: https://client.example.com/cb#error=access_denied&state=xyz
```

3.4.3. 密码模式(Resource Owner Password Credentials Grant)

密码模式适合于资源所有者与客户端具有信任关系的情况,如设备操作系统或高级特权应用.当启用这种许可类型时授权服务器应该特别关照且只有当其他流程都不可用时才可以.

这种许可类型适合于能够获得资源所有者凭据(用户名和密码,通常使用交互的形式)的客户端.通过转换已存储的凭据至访问令牌,它也用于迁移现存的使用如 HTTP 基本或摘要身份验证的直接身份验证方案的客户端至 OAuth.

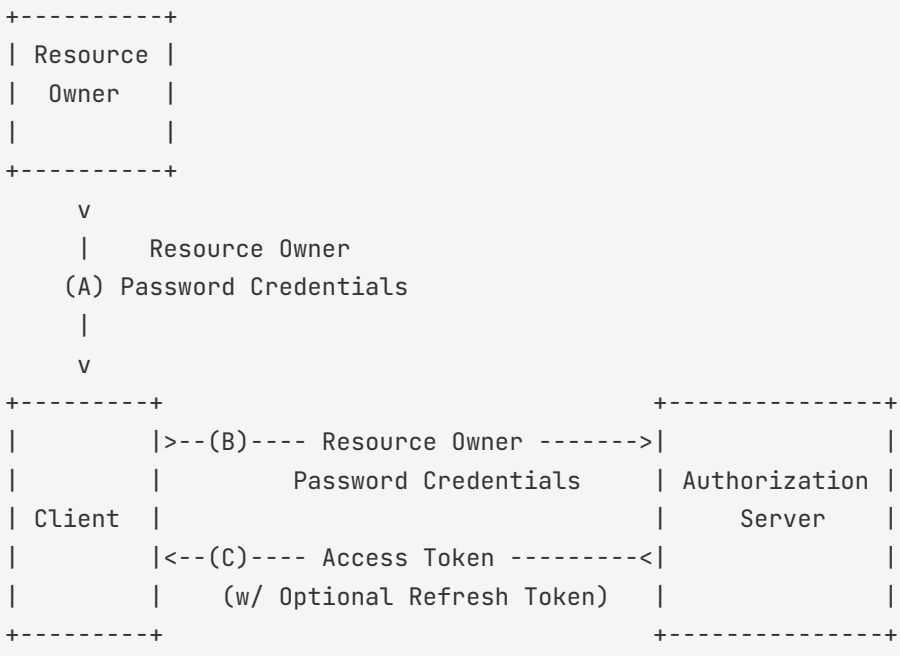


Figure 5: Resource Owner Password Credentials Flow

图5所示的流程包括以下步骤:

A. 资源所有者向客户端提供其用户名和 密码.

B. 通过包含从资源所有者处接收到的凭据,客户端从授权服务器的令牌端点请求访问令牌.当发起请求时,客户端与授权服务器进行身份验证.

C. 授权服务器对客户端进行身份验证,验证资源所有者的凭证,如果有效,颁发访问令牌.

授权请求和响应(Authorization Request and Response)

客户端获得资源所有者凭据所通过的方式超出了本规范的范围.一旦获得访问令牌,客户端必须丢弃凭据.

访问令牌请求(Access Token Request)

客户端通过使用按 附录B `application/x-www-form-urlencoded` 格式在 HTTP 请求实体正文中发送下列 UTF-8 字符编码的参数向令牌端点发起请求:

`grant_type`

必需的.值必须设置为 `password`.

username

必需的.资源所有者的用户名.

password

必需的.资源所有者的密码.

scope

可选的.如如第 3.3 节所述的访问请求的范围.

如果客户端类型是机密的或向客户端颁发了客户端凭据(或选定的其他身份验证要求),客户端必须按照第 3.2.1 节 所述与授权服务器进行身份验证.

例如,客户端使用传输层安全发起如下HTTP请求(额外的换行仅用于显示目的):

```
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
```

授权服务器必须:

- 要求机密客户端或任何被颁发了客户端凭据(或有其他身份验证要求)的客户端进行客户端身份验证,
- 若包括了客户端身份验证,验证客户端身份,并
- 使用它现有的密码验证算法验证资源所有者的密码凭据.

由于这种访问令牌请求使用了资源所有者的密码,授权服务器必须保护端点防止暴力攻击(例如,使用速率限制或生成警报).

访问令牌响应(Access Token Response)

如果访问令牌请求有效且已授权,则授权服务器发出访问令牌和可选的刷新令牌,如 第 5.1 节所述. 如果请求客户端身份验证失败或无效,授权服务器返回 如第 5.2 节所述的错误响应.

成功响应的示例:

```
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"
}
```

3.4.4. 客户端模式(Client Credentials Grant)

当客户端请求访问它所控制的,或者事先与授权服务器协商(所采用的方法超出了本规范的范围)的其他资源所有者的受保护资源,客户端可以只使用它的客户端凭据(或者其他受支持的身份验证方法)请求访问令牌.

客户端凭据许可类型必须只能由机密客户端使用.

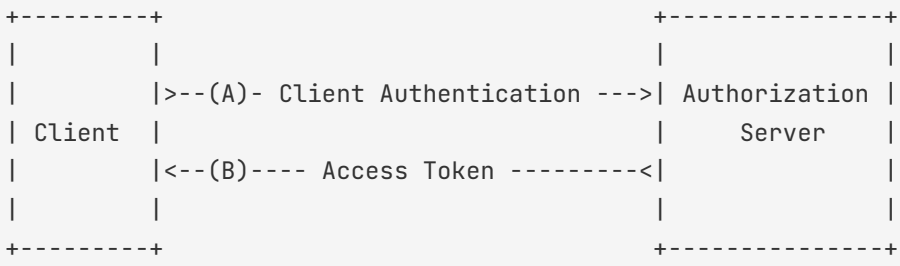


图 6: Client Credentials Flow

图6所示的流程包括以下步骤:

A. 客户端与授权服务器进行身份验证并向令牌端点请求访问令牌.

B. 授权服务器对客户端进行身份验证,如果有效,颁发访问令牌.

授权请求和响应(Authorization Request and Response)

由于客户端身份验证被用作授权许可,所以不需要其他授权请求.

访问令牌请求(Access Token Request)

客户端通过使用按 附录B application/x-www-form-urlencoded 格式在 HTTP 请求实体正文中发送下列 UTF-8 字符编码的参数向令牌端点发起请求:

grant_type

必需的.值必须设置为 client_credentials.

scope

可选的.如如第 3.3 节所述的访问请求的范围.

客户端必须按照第 3.2.1 节 所述与授权服务器进行身份验证.

例如,客户端使用传输层安全发起如下 HTTP 请求(额外的换行仅用于显示目的):

```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials
```

授权服务器必须对客户端进行身份验证.

访问令牌响应(Access Token Response)

如果访问令牌请求是有效的且被授权,授权服务器如 第 5.1 节所述 颁发访问令牌 .刷新令牌不应该包含在内. 如果请求因客户端身份验证失败或无效,授权服务器如第 5.2 节所述的返回错误响应.

一个样例成功响应:

```
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"
}
```

3.4.5. 扩展授权

通过使用绝对 URI 作为令牌端点的 grant_type 参数的值指定许可类型,并通过添加任何其他需要的参数,客户端使用扩展许可类型.

例如,采用 OAuth-SAML 定义的安全断言标记语言(SAML)2.0 断言许可类型请求访问令牌,客户端可以使用 TLS 发起如下的 HTTP 请求(额外的换行仅用于显示目的):

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Asaml2-
bearer&assertion=PEFzc2VydGlvbiBJc3N1ZUluc3RhbnQ9IjIwMTEtMDU
[...omitted for brevity...]aG5TdGF0ZW1lbnQ-PC9Bc3NlcnRpb24-
```

如果访问令牌请求有效且已授权,则授权服务器发出访问令牌和可选的刷新令牌,如 第 5.1 节所述. 如果请求客户端身份验证失败或无效,授权服务器返回 如第 5.2 节所述的错误响应.

3.5. 颁发令牌

如果访问令牌请求有效且已授权,则授权服务器发出访问令牌和可选的刷新令牌,如 第 5.1 节所述. 如果请求客户端身份验证失败或无效,授权服务器返回 如第 5.2 节所述的错误响应.

3.5.1. 成功响应

授权服务器颁发访问令牌和可选的刷新令牌,通过向 HTTP 响应实体正文中添加下列参数并使用

200(OK)状态码构造响应:

access_token

必需的.授权服务器颁发的访问令牌.

token_type

必需的.如 7.1 节所述所述的颁发的令牌的类型.值是大小写不敏感的.

expires_in

推荐的.以秒为单位的访问令牌生命周期.例如,值 3600 表示访问令牌将在从生成响应时的 1 小时后到期.如果省略,则授权服务器应该通过其他方式提供过期时间,或者记录默认值.

refresh_token

可选的.刷新令牌,可以用于如 第 6 节所述使用相同的授权许可获得新的访问令牌.

scope

可选的,若与客户端请求的范围相同; 否则,必需的.如第 3.3 节所述的访问令牌的范围.

这些参数使用 RFC4627 定义的 application/json 媒体类型包含在 HTTP 响应实体正文中.通过将每个参数添加到最高结构级别, 参数被序列化为 JavaScript 对象表示法(JSON)的结构.参数名称和字符串值作为 JSON 字符串类型包含.数值的值作为 JSON 数字类型包含.参数顺序无关并可以变化.

在任何包含令牌、凭据或其他敏感信息的响应中,授权服务器必须在其中包含值为 "no-store" 的HTTP "Cache-Control" 响应头部域 RFC2616,和值为 "no-cache" 的 "Pragma" 响应头部域 RFC2616.

例如:

```
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"
}
```

客户端必须忽略响应中不能识别的值的名称.令牌和从授权服务器接收到的值的大小未定义.客户端应该避免对值的大小做假设.授权服务器应记录其发放的任何值的大小.

3.5.2. 失败响应

授权服务器使用 HTTP 400(错误请求)状态码响应,在响应中包含下列参数:

- error: 必需的 一个单个 ASCII USASCII 错误码.值域如下
  - invalid_request : 请求缺少必需的参数、包含不支持的参数值(除了许可类型)、重复参数、包含多个凭据、采用超过一种客户端身份验证机制或其他不规范的格式.
  - invalid_client : 客户端身份验证失败(例如,未知的客户端,不包含客户端身份验证,或不支持的身份验证方法).授权服务器可以返回 HTTP 401(未授权)状态码来指出支持的 HTTP 身份验证方案.如果客户端试图通过 Authorization 请求头域进行身份验证,授权服务器必须响应 HTTP 401(未授权)状态码,并包含与客户端使用的身份验证方案匹配的 WWW-Authenticate 响应头字段.
  - invalid_grant : 提供的授权许可(如授权码、资源所有者凭据)或刷新令牌无效、过期、吊销、与在授权请求使用的重定向URI不匹配或颁发给另一个客户端.
  - unauthorized_client : 进行身份验证的客户端没有被授权使用这种授权许可类型.
  - unsupported_grant_type : 授权许可类型不被授权服务器支持.

- invalid_scope :
  请求的范围无效、未知的、格式不正确或超出资源所有者许可的范围.

  "error" 参数的值不能包含集合 %x20-21 / %x23-5B / %x5D-7E 以外的字符.

- error_description: 可选的. 提供错误描述的其他信息
  ,用于协助客户端开发人员了解发生的错误. "error_description"
  参数的值不能包含集合 %x20-21 / %x23-5B / %x5D-7E 的字符.

- error_uri:可选的. 提供一个错误页面
  ,帮助客户端开发人员了解更多有关错误的其他信息. "error_uri" 参数的值必须符合
  URI-reference 语法,因此不能包含集合 %x21 / %x23-5B / %x5D-7E 以外的字符.

这些参数使用 RFC4627 定义的 application/json 媒体类型包含在HTTP响应实体正文中
.通过将每个参数添加到最高结构级别, 参数被序列化为 JavaScript 对象表示法
(JSON)的结构. 参数名称和字符串值作为 JSON 字符串类型包含.数值的值作为 JSON
数字类型包含.参数顺序无关并可以变化.

例如:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  "error":"invalid_request"
}
```

3.6. 刷新令牌

若授权服务器给客户端颁发了刷新令牌,客户端通过使用按 附录B application/x-www-form-urlencoded 格式在 HTTP 请求实体正文中发送下列 UTF-8
字符编码的参数向令牌端点发起刷新请求:

grant_type

必需的.值必须设置为 refresh_token.

refresh_token

必需的.颁发给客户端的刷新令牌.

scope

可选的.如第 3.3 节所述的访问请求的范围
.请求的范围不能包含任何不是由资源所有者原始许可的范围,若省略,被视为与资源所有者原始许可的范围相同.

因为刷新令牌通常是用于请求额外的访问令牌的持久凭证,刷新令牌绑定到被它被颁发给的客户端.如果客户端类型是机密的或客户端被颁发了客户端凭据(或选定的其他身份验证要求),客户端必须如 第 3.2.1 节 节所述与授权服务器进行身份验证.

例如,客户端使用传输层安全发起如下 HTTP 请求(额外的换行仅用于显示目的):

```
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
```

授权服务器必须:

- 要求机密客户端或任何被颁发了客户端凭据(或有其他身份验证要求
)的客户端进行客户端身份验证,
- 若包括了客户端身份验证
  ,验证客户端身份并确保刷新令牌是被颁发给进行身份验证的客户端的,并
- 验证刷新令牌.

如果有效且被授权,授权服务器如 第 5.1 节所述 颁发访问令牌.如果请求因验证失败或无效,授权服务器 如第 5.2 节所述 返回错误响应.

授权服务器可以颁发新的刷新令牌,在这种情况下,客户端必须放弃旧的刷新令牌,替换为新的刷新令牌.在向客户端颁发新的刷新令牌后授权服务器可以撤销旧的刷新令牌.若颁发了新的刷新令牌,刷新令牌的范围必须与客户端包含在请求中的刷新令牌的范围相同.

3.7. 访问受保护的资源

通过向资源服务器出示访问令牌,客户端访问受保护资源.资源服务器必须验证访问令牌,并确保它没有过期且其范围涵盖了请求的资源.资源服务器用于验证访问令牌的方法(以及任何错误响应)超出了本规范的范围,但一般包括资源服务器和授权服务器之间的互动或协调.

客户端使用访问令牌与资源服务器进行证认的方法依赖于授权服务器颁发的访问令牌的类型.通常,它涉及到使用具有所采用的访问令牌类型的规范定义的身份验证方案(如 RFC6750)的HTTP Authorization 的请求头字段 RFC2617.

3.7.1. 访问令牌类型

访问令牌的类型给客户端提供了成功使用该访问令牌(和类型指定的属性)发起受保护资源请求所需的信息 若客户端不理解令牌类型,则不能使用该访问令牌.

例如,https://tools.ietf.org/html/rfc6750[RFC6750] 定义的 "bearer" 令牌类型简单的在请求中包含访问令牌字符串来使用:

```
GET /resource/1 HTTP/1.1
Host: example.com
Authorization: Bearer mF_9.B5f-4.1JqM
```

而 OAuth-HTTP-MAC 定义的 "mac" 令牌类型通过与许可类型一起颁发用于对 HTTP 请求中某些部分签名的消息认证码(MAC)的密钥来使用.

```
GET /resource/1 HTTP/1.1
Host: example.com
Authorization: MAC id="h480djs93hd8",
                   nonce="274312:dj83hs9s",
                   mac="kDZvddkndxvhGRXZhvuDjEWhGeE="
```

提供上面的例子仅作说明用途.建议开发人员在使用前查阅 RFC6750 和 OAuth-HTTP-MAC 规范.

每一种访问令牌类型的定义指定与 access_token 响应参数一起发送到客户端的额外属性 .它还定义了HTTP验证方法当请求受保护资源时用于包含访问令牌.

3.7.2. 错误响应

如果资源访问请求失败,资源服务器应该通知客户端该错误.虽然规定这些错误响应超出了本规范的范围,但是本文档在 11.4 节建立了一张公共注册表,用作 OAuth 令牌身份验证方案之间分享的错误值.

主要为 OAuth 令牌身份验证设计的新身份验证方案应该定义向客户端提供错误状态码的机制,其中允许的错误值限于本规范建立的错误注册表中.

这些方案可以限制有效的错误代码是注册值的子集.如果错误代码使用命名参数返回,该参数名称应该是 `error`.

其他能够被用于 OAuth 令牌身份验证的方案,但不是主要为此目的而设计的,可以帮顶他们的错误值到相同方式的注册表项.

新的认证方案也可以选择指定使用 `error_description` 和 `error_uri` 参数,用于以本文档中用法相同的方式的返回错误信息.

3.8. 可扩展性

3.8.1. 定义访问令牌类型

访问令牌类型可以用以下两种方法之一来定义:在访问令牌类型注册表中注册(按 11.1 节中的过程)的,或者通过使用一个唯一的绝对 URI 作为它的名字.

采用 URI 命名的类型应该限定于特定供应商的实现,它们不是普遍适用的并且特定于使用它们的资源服务器的实现细节.

所有其他类型都必须注册.类型名称必需符合 type-name ANBF.如果类型定义包含了一种新的 HTTP 身份验证方案,该类型名称应该与该 HTTP 身份验证方案名称一致(如 RFC2617 定义).令牌类型 "example" 被保留用于样例中.

```
type-name  = 1*name-char
name-char  = "-" / "." / "_" / DIGIT / ALPHA
```

3.8.2. 定义新的端点参数

用于授权端点或令牌端点的新的请求或响应参数按照11.2 节 中的过程在 OAuth 参数注册表中定义和注册.

参数名称必须符合 param-name ABNF,并且参数值的语法必须是明确定义的(例如,使用 ABNF,或现有参数的语法的引用).

```
param-name  = 1*name-char
name-char   = "-" / "." / "_" / DIGIT / ALPHA
```

不是普遍适用的并且特定于使用它们的授权服务器的实现细节的未注册的特定供应商的参数扩展应该采用特定供应商的前缀(例如,以 'companyname_' 开头),从而不会与其他已注册的值冲突.

3.8.3. 定义新的授权类型

新的授权许可类型可以通过赋予它们一个 grant_type 参数使用的唯一的绝对URI来定义 .如果扩展许可类型需要其他令牌端点参数,它们必须如 11.2 节 所述在 OAuth 参数注册表中注册.

3.8.4. 定义新的授权端点响应类型

用于授权端点的新的响应类型按照 11.3 节中的过程在授权端点响应类型注册表中定义和注册 .响应类型名称必须符合 response-type ABNF.

```
response-type  = response-name *( SP response-name )
response-name  = 1*response-char
response-char  = "_" / DIGIT / ALPHA
```

如果响应类型包含一个或多个空格字符 (%x20),它被看作是一个空格分隔的值列表 ,其中的值的顺序不重要.只有一种值的顺序可以被注册,它涵盖了相同的值的集合的所有其他排列 .

例如,响应类型 "token code" 未由本规范定义.然而,一个扩展可以定义和注册 "token code" 响应类型. 一旦注册,相同的组合 "code token" 不能被注册 ,但是这两个值都可以用于表示相同的响应类型.

3.8.5. 定义其他错误代码

在协议扩展(例如,访问令牌类型、扩展参数或扩展许可类型等)需要其他错误代码用于授权码模式错误响应(如第 4.1.2.1 节)、简化模式错误响应(4.2.2.1 节)、令牌错误响应(5.2 节)或资源访问错误响应(7.2 节)的情况下,这些错误代码可以被定义.

如果用于与它们配合的扩展是已注册的访问令牌类型,已注册的端点参数或者扩展许可类型,扩展错误代码必须被注册.用于未注册扩展的错误代码可以被注册.

错误代码必须符合的 `error` ABNF,且可能的话应该以一致的名称作前缀.例如,一个表示给扩展参数 `"example"` 设置了无效值的错误应该被命名为 `"example_invalid"`.

```
error      = 1*error-char
error-char = %x20-21 / %x23-5B / %x5D-7E
```

3.9. 本地应用程序(Native Applications)

本地应用程序是安装和执行在资源所有者使用的设备上的客户端(例如,桌面程序,本地移动应用).本地应用程序需要关于安全、平台能力和整体最终用户体验的特别注意事项.

授权端点需要在客户端和资源所有者用户代理之间进行交互.本地应用程序可以调用外部的用户代理,或在应用程序中嵌入用户代理.例如:

- 外部用户代理-本地应用程序可以捕获来自授权服务器的响应.它可以使用带有操作系统已注册方案的重定向URI调用客户端作为处理程序,手动复制粘贴凭据,运行本地Web服务器,安装用户代理扩展,或者通过提供重定向URI来指定客户端控制下的服务器托管资源,反过来使响应对本地应用程序可用.
- 嵌入式用户代理-通过监视资源加载过程中发生的状态变化或者访问用户代理的cookies存储,本地应用程序直接与嵌入式用户代理通信,获得响应.

当在外部或嵌入式用户代理中选择时,开发者应该考虑如下:

- 外部用户代理可能会提高完成率,因为资源所有者可能已经有了与授权服务器的活动会话,避免了重新进行身份验证的需要.它提供了熟悉的最终用户体验和功能.资源所有者可能也依赖于用户代理特性或扩展帮助他进行身份验证(例如密码管理器、两步设备读取器)

- 嵌入式用户代理可能会提供更好的可用性,因为它避免了切换上下文和打开新窗口的需要.
- 嵌入式用户代理构成了安全挑战,因为资源所有者在一个未识别的窗口中进行身份验证,无法获得在大多数外部用户代理中的可视的保护.嵌入式用户代理教育用户信任未标识身份验证请求(使钓鱼攻击更易于实施).

当在简化模式和授权码模式中选择时,下列应该被考虑:

- 使用授权码模式类型的本地应用程序应该这么做而不需使用用户凭据,因为本地应用程序无力保持客户端凭据的机密性.
- 当使用简化模式类型流程时,刷新令牌不会返回,这就要求一旦访问令牌过期就要重复授权过程.

3.10. 安全注意事项

作为一个灵活的可扩展的框架,OAuth 的安全性考量依赖于许多因素.
以下小节提为实现者提供了聚焦在 2.1 节所述的三种客户端配置上的安全指南:Web 应用、基于用户代理的应用和本地应用程序.

全面的 OAuth 安全模型和分析以及该协议设计的背景在 OAuth-THREATMODE 中提供.

3.10.1. 客户端认证(Client Authentication)

授权服务器为进行客户端身份验证的目的,为 Web 应用客户端创建客户端凭据.授权服务器被鼓励考虑比客户端密码更强的客户端身份验证手段.Web 应用程序客户端必须确保客户端密码和其他客户端凭据的机密性.

授权不得向本地应用程序或基于用户代理的应用客户端颁发客户端密码或其他客户端凭据用于客户端验证目的.授权服务器可以颁发客户端密码或其他凭据给专门的设备上特定安装的本地应用程序客户端.

当客户端身份验证不可用时,授权服务器应该采用其他方式来验证客户端的身份-例如,通过要求客户端重定向 URI 的注册或者引入资源所有者来确认身份.当请求资源所有者授权时,有效的重定向URI是不足以验证客户端的身份,但可以用来防止在获得资源所有者授权后将凭据传递给假冒的客户端.

授权服务器必须考虑与未进行身份验证的客户端交互的安全实现并采取措施限制颁发给这些客户

端的其他凭据(如刷新令牌)的潜在泄露.

3.10.2. 客户端假冒(Client Impersonation)

如果被仿冒的客户端不能,或无法保持其客户端凭据保密.恶意客户端可能冒充其他客户端,并获得对受保护资源的访问权限.

授权服务器任何可能的时候必须验证客户端身份.如果授权服务器由于客户端的性质无法对客户端进行身份验证,授权服务器必须要求注册任何用于接收授权响应的重定向URI并且应该利用其他手段保护资源所有者防止这样的潜在仿冒客户端.例如,授权服务器可以引入资源所有者来帮助识别客户端和它的来源.

授权服务器应该实施显式的资源所有者身份验证并且提供给资源所有者有关客户端及其请求的授权范围和生命周期的信息.由资源所有者在当前客户端上下文中审查信息并授权或拒绝该请求.

授权服务器未对客户端进行身份验证(没有活动的资源所有者交互)或未依靠其他手段确保重复的请求来自于原始客户端而非冒充者时,不应该自动处理重复的授权请求.

3.10.3. 访问令牌(Access Tokens)

访问令牌凭据(以及任何机密的访问令牌属性)在传输和储存时必须保持机密性,并只与授权服务器、访问令牌生效的资源服务器和访问令牌被颁发的客户端共享.访问令牌凭据必须只能使用带有 RFC2818 定义的服务器身份验证的 1.6 节所述的 TLS 传输.

当使用简化模式授权许可类型时,访问令牌在 URI 片段中传输
,这可能泄露访问令牌给未授权的一方.

授权服务器必须确保访问令牌不能被生成、修改或被未授权一方猜测而产生有效的访问令牌.

客户端应该为最小范围的需要请求访问令牌.授权服务器在选择如何兑现请求的范围时应该将客户端身份考虑在内,且可以颁发具有比请求的更少的权限的访问令牌.

本规范未给资源服务器提供任何方法来确保特定的客户端提交给它的访问令牌是授权服务器颁发给此客户端的.

3.10.4. 刷新令牌(Refresh Tokens)

授权服务器可以给Web应用客户端和本地应用程序客户端颁发刷新令牌.

刷新令牌在传输和储存时必须保持机密性,并只与授权服务器和刷新令牌被颁发的客户端共享.授权服务器必须维护刷新令牌和它被颁发给的客户端之间的绑定.刷新令牌必须只能使用带有 RFC2818 定义的服务器身份验证的 1.6 节 所述的TLS 传输.
授权服务器必须验证刷新令牌和客户端身份之间的绑定,无论客户端身份是否能被验证.当无法进行客户端身份验证时,授权服务器应该采取其他手段检测刷新令牌滥用.

例如,授权服务器可以使用刷新令牌轮转机制,随着每次访问令牌刷新响应,新的刷新令牌被颁发.以前的刷新令牌被作废但是由授权服务器保留.如果刷新令牌被泄露,随后同时被攻击者和合法客户端使用,他们中一人将提交被作废的刷新令牌,这将通知入侵给授权服务器.

授权服务器必须确保刷新令牌不能被生成、修改或被未授权一方猜测而产生有效的刷新令牌.

3.10.5. 授权码(Authorization Codes)

授权码的传输应该建立在安全通道上,客户端应该要求在它的重定向URI上使用TLS,若该URI指示了一个网络资源. 由于授权码由用户代理重定向传输,它们可能潜在地通过用户代理历史记录和 HTTP 参照 header 被泄露.

授权码明以纯文本承载凭据使用,用于验证在授权服务器许可权限的资源所有者就是返回到客户端完成此过程的相同的资源所有者.因此,如果客户端依赖于授权码作为它自己的资源所有者身份验证,客户端重定向端点必须要求使用 TLS.

授权码必须是短暂的且是单用户的.如果授权服务器观察到多次以授权码交换访问令牌的尝试,授权服务器应该试图吊销所有基于泄露的授权码而颁发的访问令牌.

如果客户端可以进行身份验证,授权服务器必须验证客户端身份,并确保授权码颁发给了同一个客户端.

3.10.6. 授权码重定向URI操作

当使用授权码许可类型请求授权时,客户端可以通过 redirect_uri 参数指定重定向 URI.
如果攻击者能够伪造重定向URI的值,这可能导致授权服务器向攻击者控制的 URI 重定向带有授权码的资源所有者用户代理.

攻击者可以在合法客户端上创建一个帐户,并开始授权流程.当攻击者的用户代理被发送到授权服务器来许可访问权限时,攻击者抓取合法客户端提供的授权 URI 并用攻击者控制下的 URI 替换客户端的重定向 URI.

攻击者然后欺骗受害者顺着仿冒的链接来对合法客户端授权访问权限.

一旦在授权服务器—受害者被唆使代表一个合法的被信任的客户端使用正常有效的请求—授权该请求时.受害者然后带着授权码重定向到受攻击者控制的端点.通过使用客户端提交的原始重定向 URI 向客户端发送授权码,攻击者完成授权流程
.客户端用授权码交换访问令牌并与将它与攻击者的客户端账号关联,该账户现在能获得受害者授权的(通过客户端)对访问受保护资源的访问权限.

为了防止这种攻击,授权服务器必须确保用于获得授权码的重定向 URI
与当用授权码交换访问令牌时提供的重定向 URI 相同.授权服务器必须要求公共客户端
,并且应该要求机密客户注册它们的重定向 URI.如果在请求中提供一个重定向
URI,授权服务器必须验证对注册的值.如果在请求中提供了重定向
URI,授权服务器必须对比已注册的.

3.10.7. 密码(Resource Owner Password Credentials)

资源所有者密码凭据许可类型通常用于遗留或迁移原因.它降低了由客户端存储用户名和密码的整体风险,但并没有消除泄露高度特权的凭证给客户端的需求.

这种许可类型比其他许可类型承载了更高的风险,因为它保留了本协议寻求避免的密码反模式.客户端可能滥用密码或密码可能会无意中被泄露给攻击者(例如,通过客户端保存的日志文件或其他记录).

此外,由于资源拥有者对授权过程没有控制权(在转手它的凭据给客户端后资源所有者的参与结束)
,客户端可以获得比资源所有者预期的具有更大范围的访问令牌.授权服务器应该考虑由这种许可类型颁发的访问令牌的范围和寿命.

授权服务器和客户端应该尽量减少这种许可类型的使用,并尽可能采用其他许可类型.

3.10.8. 请求加密(Request Confidentiality)

访问令牌、刷新令牌、资源所有者密码和客户端凭据不能以明文传输.授权码不应该以明文传输.

state 和 scope 参数不应该包含敏感的客户端或资源所有者的纯文本信息
,因为它们可能在不安全的通道上被传输或被不安全地存储.

3.10.9. 确保端点真实性

为了防止中间人攻击,授权服务器必须对任何被发送到授权和令牌端点的请求要求 RFC2818 中定义的具有服务器身份验证的 TLS 的使用.客户端必须按 RFC6125 定义且按照它服务器身份进行身份验证的需求验证授权服务器的的 TLS 证书.

3.10.10. 凭证猜测攻击

授权服务器必须防止攻击者猜测访问令牌、授权码、刷新令牌、资源所有者密码和客户端凭据.

攻击者猜测已生成令牌(和其它不打算被最终用户掌握的凭据)的概率必须小于或等于 2^(-128),并且应该小于或等于 2^(-160).

授权服务器必须采用其他手段来保护打算给最终用户使用的凭据.

3.10.11. 网络钓鱼攻击

本协议或类似协议的广泛部署,可能导致最终用户变成习惯于被重定向到要求输入他们的密码的网站的做法.

如果最终用户在输入他们的凭据前不注意辨别这些网站的真伪,这将使攻击者利用这种做法窃取资源所有者的密码成为可能.

服务提供者应尝试教育最终用户有关钓鱼攻击构成的风险,并且应该为最终用户提供使确认它们的站点的真伪变得简单的机制.客户端开发者应该考虑他们如何与用户代理(例如,外部的和嵌入式的)交互的安全启示以及最终用户辨别授权服务器真伪的能力.

为了减小钓鱼攻击的风险,授权服务器必须要求在用于最终用户交互的每个端点上使用 TLS.

3.10.12. 跨站请求伪造

跨站请求伪造(CSRF)是一种漏洞利用,攻击者致使受害的最终用户按恶意 URI(例如以误导的链接、图片或重定向提供给用户代理)到达受信任的服务器(通常由存在有效的会话 Cookie 而建立).

针对客户端的重定向 URI 的 CSRF 攻击允许攻击者注入自己的授权码或访问令牌,这将导致在客户端中使用与攻击者的受保护资源关联的访问令牌而非受害者的(例如,保存受害者的银行账户信息到攻击者控制的受保护资源).

客户端必须为它的重定向 URI 实现 CSRF 保护.这通常通过要求向重定向 URI
端点发送的任何请求包含该请求对用户代理身份认证状态的绑定值(例如,用于对用户代理进行身份验证的会话 Cookie 的哈希值)来实现.客户端应该使用
"state”请求参数在发起授权请求时向授权服务器传送该值.

一旦从最终用户获得授权,授权服务器重定向最终用户的用户代理带着要求的包含在 state
参数中的绑定值回到客户端. 通过该绑定值与用户代理的身份验证状态的匹配
,绑定值使客户端能够验证请求的有效性.用于CSRF保护的绑定值必须包含不可猜测的值(如
10.10 节所述)且用户代理的身份验证状态(例如会话 Cookie、HTML5 本地存储
)必须保存在只能被客户端和用户代理访问的地方(即通过同源策略保护).

针对授权服务器的授权端点的 CSRF
攻击可能导致攻击者获得最终用户为恶意客户端的授权而不牵涉或警告最终用户.

授权服务器必须为它的授权端点实现 CSRF
保护并且确保在资源所有者未意识到且无显式同意时恶意客户端不能获得授权.

3.10.13. 点击劫持

在点击劫持攻击中,攻击者注册一个合法客户端然后构造一个恶意站点,在一个透明的覆盖在一组虚假按钮上面的嵌入框架中加载授权服务器的授权端点Web页面,这些按钮被精心构造恰好放置在授权页面上的重要按钮下方.当最终用户点击了一个误导的可见的按钮时,最终用户实际上点击了授权页面上一个不可见的按钮(例如 "Authorize" 按钮).
这允许攻击者欺骗资源所有者许可它的客户端最终用户不知晓的访问权限.

为了防止这种形式的攻击,在请求最终用户授权时本地应用程序应该使用外部浏览器而非应用程序中嵌入的浏览器. 对于大多数较新的浏览器,避免嵌入框架可以由授权服务器使用(非标准的)x-frame-options 请求头实现. 该请求头可以有两个值,deny 和 sameorigin
,它将阻止任何框架,或按不同来源的站点分别构造框架. 对于较旧的浏览器,JavaScript
框架破坏技术可以使用,但可能不会在所有的浏览器中生效.

3.10.14. 代码注入和输入验证

代码注入攻击当程序使用的输入或其他外部变量未清洗而导致对程序逻辑的修改时发生.
这可能允许攻击者对应用程序的设备或它的数据的访问权限,导致服务拒绝或引入许多的恶意副作用.

授权服务器和客户端必须清洗(并在可能的情况下验证)收到的任何值—特别是,`state` 和 `redirect_uri` 参数的值.

3.10.15. 自由重定向器(Open Redirectors)

授权服务器、授权端点和客户端重定向端点可能被不当配置,被作为自由重定向器.自由重定向器是一个使用参数自动地向参数值指定而无任何验证的地址重定向用户代理的端点.

自由重定向器可被用于钓鱼攻击,或者被攻击者通过使用熟悉的受信任的目标地址的URI授权部分使最终用户访问恶意站点.此外,如果授权服务器允许客户端只注册部分的重定向
URI,攻击者可以使用客户端操作的自由重定向器构造重定向
URI,这将跳过授权服务器验证但是发送授权码或访问令牌给攻击者控制下的端点.

3.10.16. 在简化模式中滥用访问令牌来假冒资源所有者

对于使用简化模式的公共客户端,本规范没有为客户端提供任何方法来决定访问令牌颁发给的是什么样的客户端.

资源所有者可能通过给攻击者的恶意客户端许可访问令牌自愿委托资源的访问权限.这可能是由于钓鱼或一些其他借口.攻击者也可能通过其他机制窃取令牌.
攻击者然后可能会尝试通过向合法公开客户端提供该访问令牌假冒资源拥有者.

在简化模式流程 (`response_type=token`) 中
,攻击者可以轻易转换来自授权服务器的响应中的令牌,用事先颁发给攻击者的令牌替换真实的访问令牌.

依赖于在返回通道中传递访问令牌识别客户端用户的与本地应用程序通信的服务器可能由攻击者创建能够注入随意的窃取的访问令牌的危险的程序被类似地危及.

任何做出只有资源所有者能够提交给它有效的为资源的访问令牌的假设的公共客户端都是易受这种类型的攻击的.

这种类型的攻击可能在合法的客户端上泄露有关资源所有者的信息给攻击者(恶意客户端).这也将允许攻击者在合法客户端上用和资源所有者相同的权限执行操作,该资源所有者最初许可了访问令牌或授权码.

客户端对资源拥有者进行身份验证超出了本规范的范围.任何使用授权过程作为客户端对受委托的最终用户进行身份验证的形式的规范(例如,第三方登录服务)不能在没有其他的客户端能够判断访

问令牌是否颁发是颁发给它使用的安全机制的情况下使用简化模式流程(例如,限制访问令牌的受众).

3.11. IANA 注意事项

3.11.1. OAuth 访问令牌类型注册表.

本规范建立 OAuth 访问令牌类型注册表.

在 oauth-ext-review@ietf.org 邮件列表上的两周的审查期后
,根据一位或多位指定的专家的建议下,按规范需求(RFC5226)注册访问令牌类型.然而,为允许发表之前的值的分配,指定的专家(们)一旦他们对这样的规范即将发布感到满意可以同意注册.

注册请求必须使用正确的主题(例如"访问令牌类型 "example" 的请求)发送到 oauth-ext-review@ietf.org 邮件列表来审查和评论.

在审查期间,指定的专家(们)将同意或拒绝该注册请求,向审查列表和 IANA 通报该决定
.拒绝应该包含解释,并且可能的话,包含如何使请求成功的建议.

IANA必须只接受来自指定的专家(们)的注册表更新并且应该引导所有注册请求至审查邮件列表.

注册模板

Type name

请求的名称(例如,"example").

Additional Token Endpoint Response Parameters

随 "access_token" 参数一起返回的其他响应参数. 新的参数都必须如 11.2 节所述在 OAuth 参数注册表中分别注册.

HTTP Authentication Scheme(s)

HTTP 身份验证方案名称,如果有的话
,用于使用这种类型的访问令牌对受保护资源进行身份验证.

Change controller

对于标准化过程的 RFC,指定为 "IETF". 对于其他,给出负责的部分的名称. 其他细节

(例如,邮政地址,电子邮件地址,主页URI)也可以包括在内.

Specification document(s)

指定参数的文档的引用文献,最好包括可以用于检索文档副本的URI.
相关章节的指示也可以包含但不是必需的.

3.11.2. OAuth 参数注册表

本规范建立OAuth参数注册表.

在 oauth-ext-review@ietf.org 邮件列表上的两周的审查期后
,根据一位或多位指定的专家的建议下,按规范需求(RFC5226)注册列入授权端点请求、授权端点响应、令牌端点请求或令牌端点响应的其他参数.然而,为允许发表之前的值的分配,指定的专家(们)一旦他们对这样的规范即将发布感到满意可以同意注册.

注册请求必须使用正确的主题(例如,参数 "example" 的请求)发送到 oauth-ext-review@ietf.org 邮件列表来审查和评论.

在审查期间,指定的专家(们)将同意或拒绝该注册请求,向审查列表和IANA通报该决定.拒绝应该包含解释,并且可能的话,包含如何使请求成功的建议.

IANA必须只接受来自指定的专家(们)的注册表更新并且应该引导所有注册请求至审查邮件列表.

注册模板

Parameter name

请求的名称(例如,"example").

Parameter usage location

可以使用参数的位置.可以是授权请求、授权响应、令牌 请求或令牌响应.

Change controller

对于标准化过程的 RFC,指定为 "IETF". 对于其他,给出负责的部分的名称. 其他细节
(例如,邮政地址,电子邮件地址,主页URI)也可以包括在内.

Specification document(s)

指定参数的文档的引用文献,最好包括可以用于检索文档副本的URI.
相关章节的指示也可以包含但不是必需的.

初始注册表内容

OAuth 参数注册表中的初始内容:

- Parameter name: client_id
- Parameter usage location: authorization request, token request
- Change controller: IETF
- Specification document(s): RFC 6749
- Parameter name: client_secret
- Parameter usage location: token request
- Change controller: IETF
- Specification document(s): RFC 6749
- Parameter name: response_type
- Parameter usage location: authorization request
- Change controller: IETF
- Specification document(s): RFC 6749
- Parameter name: redirect_uri
- Parameter usage location: authorization request, token request
- Change controller: IETF
- Specification document(s): RFC 6749
- Parameter name: scope
- Parameter usage location: authorization request, authorization response, token request, token response
- Change controller: IETF

- Specification document(s): RFC 6749
- Parameter name: state
- Parameter usage location: authorization request, authorization response
- Change controller: IETF
- Specification document(s): RFC 6749
- Parameter name: code
- Parameter usage location: authorization response, token request
- Change controller: IETF
- Specification document(s): RFC 6749
- Parameter name: error_description
- Parameter usage location: authorization response, token response
- Change controller: IETF
- Specification document(s): RFC 6749
- Parameter name: error_uri
- Parameter usage location: authorization response, token response
- Change controller: IETF
- Specification document(s): RFC 6749
- Parameter name: grant_type
- Parameter usage location: token request
- Change controller: IETF
- Specification document(s): RFC 6749
- Parameter name: access_token
- Parameter usage location: authorization response, token response
- Change controller: IETF

- Specification document(s): RFC 6749
- Parameter name: token_type
- Parameter usage location: authorization response, token response
- Change controller: IETF
- Specification document(s): RFC 6749
- Parameter name: expires_in
- Parameter usage location: authorization response, token response
- Change controller: IETF
- Specification document(s): RFC 6749
- Parameter name: username
- Parameter usage location: token request
- Change controller: IETF
- Specification document(s): RFC 6749
- Parameter name: password
- Parameter usage location: token request
- Change controller: IETF
- Specification document(s): RFC 6749
- Parameter name: refresh_token
- Parameter usage location: token request, token response
- Change controller: IETF
- Specification document(s): RFC 6749

3.11.3. OAuth 授权端点响应类型注册表

本规范建立 OAuth 授权端点响应类型注册表.

在 oauth-ext-review@ietf.org 邮件列表上的两周的审查期后

,根据一位或多位指定的专家的建议下,按规范需求(RFC5226)注册授权端点使用的其他响应类型.然而,为允许发表之前的值的分配,指定的专家(们)一旦他们对这样的规范即将发布感到满意可以同意注册.

注册请求必须使用正确的主题(例如"响应类型example”的请求)发送到 oauth-ext-review@ietf.org 邮件列表来审查和评论.

在审查期间,指定的专家(们)将同意或拒绝该注册请求,向审查列表和IANA通报该决定.

IANA必须只接受来自指定的专家(们)的注册表更新并且应该引导所有注册请求至审查邮件列表.

注册模板

Response type name

请求的名称(例如,"example").

Change controller

对于标准化过程的RFC,指定为 "IETF".对于其他,给出负责的部分的名称.其他细节(例如,邮政地址,电子邮件地址,主页 URI)也可以包括在内.

Specification document(s)

指定参数的文档的引用文献,最好包括可以用于检索文档副本的URI.相关章节的指示也可以包含但不是必需的

初始注册表内容

OAuth授权端点响应类型注册表的初始内容:

- Response type name: code
- Change controller: IETF
- Specification document(s): RFC 6749
- Response type name: token
- Change controller: IETF
- Specification document(s): RFC 6749

3.11.4. OAuth 扩展错误注册表

本规范建立OAuth扩展错误注册表.

在 oauth-ext-review@ietf.org 邮件列表上的两周的审查期后
,根据一位或多位指定的专家的建议下,按规范需求(RFC5226)注册与其他协议扩展(例如,扩展的许可类型、访问令牌类型或者扩展参数)一起使用的其他错误代码.然而,为允许发表之前的值的分配,指定的专家(们)一旦他们对这样的规范即将发布感到满意可以同意注册.

注册请求必须使用正确的主题(例如"错误代码example”的请求)发送到 oauth-ext-review@ietf.org 邮件列表来审查和评论.

在审查期间,指定的专家(们)将同意或拒绝该注册请求,向审查列表和IANA通报该决定.拒绝应该包含解释,并且可能的话,包含如何使请求成功的建议.

IANA必须只接受来自指定的专家(们)的注册表更新并且应该引导所有注册请求至审查邮件列表.

注册模板

Error name

请求的名称(例如,"example").错误名称的值 不能包含集合 %x20-21 / %x23-5B / %x5D-7E 以外的字符.

Error usage location

错误使用的位置.可能的位置是 如第 4.1.2.1 节)、简化模式错误响应(4.2.2.1 节)、令牌错误响应(5.2 节)或资源访问错误响应(7.2 节

Related protocol extension

与错语代码一起使用的扩展许可类型、访问令牌类型或扩展参数的名称.

Change controller

对于标准化过程的RFC,指定为 "IETF".对于其他,给出负责的部分的名称.其他细节(例如,邮政地址,电子邮件地址,主页URI)也可以包括在内.

Specification document(s)

指定参数的文档的引用文献,最好包括可以用于检索文档副本的
URI.相关章节的指示也可以包含但不是必需的.

3.12. 参考资料

3.12.1. 规范性文献(Normative References)

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119 , March 1997.
- [RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.
- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol—HTTP/1.1", RFC 2616, June 1999.
- [RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.
- [RFC4949] Shirey, R., "Internet Security Glossary, Version 2", RFC 4949, August 2007.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226 , May 2008.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, March 2011.
- [USASCII] American National Standards Institute, "Coded Character Set—7-bit American Standard Code for Information Interchange", ANSI X3.4, 1986.
- [W3C.REC-html401-19991224] Raggett, D., Le Hors, A., and I. Jacobs, "HTML 4.01 Specification", World Wide Web Consortium Recommendation REC-html401-19991224, December 1999, http://www.w3.org/TR/1999/REC-html401-19991224.
- [W3C.REC-xml-20081126] Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008, https://www.w3.org/TR/2008/REC-xml-20081126/.

3.12.2. 参考性文献(Informative References)

- [OAuth-HTTP-MAC] Hammer-Lahav, E., Ed., "HTTP Authentication: MAC Access Authentication", Work in Progress, February 2012.
- [OAuth-SAML2] Campbell, B. and C. Mortimore, "SAML 2.0 Bearer Assertion Profiles for OAuth 2.0", Work in Progress, September 2012.
- [OAuth-THREATMODEL] Lodderstedt, T., Ed., McGloin, M., and P. Hunt, "OAuth 2.0 Threat Model and Security Considerations", Work in Progress, October 2012.
- [OAuth-WRAP] Hardt, D., Ed., Tom, A., Eaton, B., and Y. Goland, "OAuth Web Resource Authorization Profiles", Work in Progress, January 2010.
- [RFC5849] Hammer-Lahav, E., "The OAuth 1.0 Protocol", RFC 5849, April 2010.

- [RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, October 2012.

3.13. 附录A.增强巴科斯-诺尔范式(ABNF)语法

本节提供了本文档中定义的元素按 RFC5234 记法的增强巴克斯诺尔范式(ABNF)的语法描述.下列 ABNF 用 Unicode 代码要点 W3C.REC-XML-20081126 的术语定义;这些字符通常以 UTF-8 编码.元素按首次定义的顺序排列.

一些定义遵循使用来自 RFC3986 "URI-reference" 的定义.

一些定义遵循使用这些通用的定义:

```
VSCHAR     = %x20-7E
NQCHAR     = %x21 / %x23-5B / %x5D-7E
NQSCHAR    = %x20-21 / %x23-5B / %x5D-7E
UNICODECHARNOCRLF = %x09 /%x20-7E / %x80-D7FF /
                    %xE000-FFFD / %x10000-10FFFF
```

(UNICODECHARNOCRLF 定义基于 W3C.REC-XML-200811262.2 节中定义的字符,但忽略了回车和换行字符.)

1. "client_id" 语法

   "client_id" 元素在 2.3.1 节定义:

   ```
   client-id     = *VSCHAR
   ```

2. "client_secret" 语法

   "client_secret" 元素在 2.3.1 节定义:

   ```
   client-secret = *VSCHAR
   ```

3. "response_type" 语法

   "response_type" 元素在 3.1.1 和 8.4 节定义:

```
response-type = response-name *( SP response-name )
response-name = 1*response-char
response-char = "_" / DIGIT / ALPHA
```

4. "scope" 语法

   "scope" 元素在 3.3 节定义:

```
scope       = scope-token *( SP scope-token )
scope-token = 1*NQCHAR
```

5. "state" 语法

   "state" 元素在 4.1.1, 4.1.2, 4.1.2.1, 4.2.1, 4.2.2, 4.2.2.1 节定义:

```
state      = 1*VSCHAR
```

6. "redirect_uri" 语法

   "redirect_uri" 元素在 4.1.1 , 4.1.3 4.2.1 节定义:

```
redirect-uri      = URI-reference
```

7. "error" 语法

   "redirect_uri" 元素在 4.1.2.1, 4.2.2.1, 5.2, 7.2, 8.5 节定义:

```
error             = 1*NQSCHAR
```

8. "error_description" 语法

   "error_description" 元素在 4.1.2.1, 4.2.2.1, 5.2, 7.2 节定义:

```
error-description = 1*NQSCHAR
```

9. "error_uri" 语法

   "error_uri" 元素在 4.1.2.1, 4.2.2.1, 5.2, 7.2 节定义:

```
error-uri         = URI-reference
```

10. "grant_type" 语法

    "grant_type" 元素在 4.1.3 4.3.2, 4.4.2, 4.5, 6 节定义:

```
grant-type = grant-name / URI-reference
grant-name = 1*name-char
name-char  = "-" / "." / "_" / DIGIT / ALPHA
```

11. "code" 语法

    "code" 元素在 4.1.3 节定义:

```
code       = 1*VSCHAR
```

12. "access_token" 语法

    "access_token" 元素在 4.2.2, 5.1 节定义:

```
access-token = 1*VSCHAR
```

13. "token_type" 语法

    "token_type" 元素在 4.2.2, 5.1, 8.1节定义:

```
token-type = type-name / URI-reference
type-name  = 1*name-char
name-char  = "-" / "." / "_" / DIGIT / ALPHA
```

14. "expires_in" 语法

"expires_in" 元素在 4.2.2, 5.1 节定义:

```
expires-in = 1*DIGIT
```

15. "username" 语法

    "username" 元素在 4.3.2 节定义:

    ```
    username = *UNICODECHARNOCRLF
    ```

16. "password" 语法

    "password" 元素在 4.3.2 节定义:

    ```
    password = *UNICODECHARNOCRLF
    ```

17. "refresh_token" 语法

    "refresh_token" 元素在 5.1,6 节定义:

    ```
    refresh-token = 1*VSCHAR
    ```

18. 端点参数语法

    新端点参数的语法在 8.2 节定义:

    ```
    param-name = 1*name-char
    name-char  = "-" / "." / "_" / DIGIT / ALPHA
    ```

3.14. 附录B:使用 application/x-www-form-urlencoded 媒体类型

在本规范公布的时候,application/x-www-form-urlencoded 媒体类型在 W3C.REC-html401-19991224 的 17.13.4 节中定义但未在 IANA MIME 媒体类型注册表

(http://www.iana.org/assignments/media-types)中注册. 此外,该定义是不完整的,因为它未考虑非 US-ASCII 的字符.

在使用这种媒体类型生成有效载荷使时为解决这个缺点,名称和值必须首先使用 UTF-8 字符编码方案 RFC3629 编码; 作为结果的八位序列然后需要使用在 W3C.REC-html401-19991224 中定义的转义规则进一步编码.

当从使用这种媒体类型的有效载荷中解析数据时,由逆向名称/值编码得到的名称和值因而需要被视作八位序列,使用UTF-8字符编码方案解码. 例如,包含六个Unicode代码点的值

```
(1) U+0020 (SPACE), (2) U+0025 (PERCENT SIGN),
(3) U+0026 (AMPERSAND), (4) U+002B (PLUS SIGN),
(5) U+00A3 (POUND SIGN), and (6) U+20AC (EURO SIGN) would be encoded
```

将被编码成如下的八位序列(使用十六进制表示):

```
20 25 26 2B C2 A3 E2 82 AC
```

然后在有效载荷中表示为:

```
+%25%26%2B%C2%A3%E2%82%AC
```

3.15. 致谢

```
The initial OAuth 2.0 protocol specification was edited by David
 Recordon, based on two previous publications: the OAuth 1.0 community
 specification [RFC5849], and OAuth WRAP (OAuth Web Resource
 Authorization Profiles) [OAuth-WRAP].  Eran Hammer then edited many
 of the intermediate drafts that evolved into this RFC.  The Security
 Considerations section was drafted by Torsten Lodderstedt, Mark
 McGloin, Phil Hunt, Anthony Nadalin, and John Bradley.  The section
 on use of the "application/x-www-form-urlencoded" media type was
 drafted by Julian Reschke.  The ABNF section was drafted by Michael
 B. Jones.
```

The OAuth 1.0 community specification was edited by Eran Hammer and authored by Mark Atwood, Dirk Balfanz, Darren Bounds, Richard M. Conlan, Blaine Cook, Leah Culver, Breno de Medeiros, Brian Eaton, Kellan Elliott-McCrea, Larry Halff, Eran Hammer, Ben Laurie, Chris Messina, John Panzer, Sam Quigley, David Recordon, Eran Sandler, Jonathan Sergent, Todd Sieling, Brian Slesinsky, and Andy Smith.

The OAuth WRAP specification was edited by Dick Hardt and authored by Brian Eaton, Yaron Y. Goland, Dick Hardt, and Allen Tom.

This specification is the work of the OAuth Working Group, which includes dozens of active and dedicated participants. In particular, the following individuals contributed ideas, feedback, and wording that shaped and formed the final specification:

Michael Adams, Amanda Anganes, Andrew Arnott, Dirk Balfanz, Aiden Bell, John Bradley, Marcos Caceres, Brian Campbell, Scott Cantor, Blaine Cook, Roger Crew, Leah Culver, Bill de hOra, Andre DeMarre, Brian Eaton, Wesley Eddy, Wolter Eldering, Brian Ellin, Igor Faynberg, George Fletcher, Tim Freeman, Luca Frosini, Evan Gilbert, Yaron Y. Goland, Brent Goldman, Kristoffer Gronowski, Eran Hammer, Dick Hardt, Justin Hart, Craig Heath, Phil Hunt, Michael B. Jones, Terry Jones, John Kemp, Mark Kent, Raffi Krikorian, Chasen Le Hara, Rasmus Lerdorf, Torsten Lodderstedt, Hui-Lan Lu, Casey Lucas, Paul Madsen, Alastair Mair, Eve Maler, James Manger, Mark McGloin, Laurence Miao, William Mills, Chuck Mortimore, Anthony Nadalin, Julian Reschke, Justin Richer, Peter Saint-Andre, Nat Sakimura, Rob Sayre, Marius Scurtescu, Naitik Shah, Luke Shepard, Vlad Skvortsov, Justin Smith, Haibin Song, Niv Steingarten, Christian Stuebner, Jeremy Suriel, Paul Tarjan, Christopher Thomas, Henry S. Thompson, Allen Tom, Franklin Tse, Nick Walker, Shane Weeden, and Skylar Woodward.

This document was produced under the chairmanship of Blaine Cook, Peter Saint-Andre, Hannes Tschofenig, Barry Leiba, and Derek Atkins. The area directors included Lisa Dusseault, Peter Saint-Andre, and Stephen Farrell.

Author’s Address

```
Dick Hardt (editor)
Microsoft
```

```
EMail: dick.hardt@gmail.com
URI:   http://dickhardt.org/
```

Chapter 4. rfc7519 - JSON Web Token (JWT)

原文链接：https://tools.ietf.org/html/rfc7519