

OAuth 2.0 授权框架

RFC6749 中文版

qunfanyi.com 翻译

Table of Contents

Introduction	1.1
说明	1.2
1. 介绍	1.3
1.1. 角色	1.3.1
1.2. 协议流程	1.3.2
1.3. 授权许可	1.3.3
1.3.1. 授权码	1.3.3.1
1.3.2. 隐式授权	1.3.3.2
1.3.3. 资源所有者密码凭据	1.3.3.3
1.3.4. 客户端凭据	1.3.3.4
1.4. 访问令牌	1.3.4
1.5. 刷新令牌	1.3.5
1.6. TLS 版本	1.3.6
1.7. Http 重定向	1.3.7
1.8. 互操作	1.3.8
1.9. 符号约定	1.3.9
2. 客户端注册	1.4
2.1. 客户端类型	1.4.1
2.2. 客户端标识	1.4.2
2.3. 客户端认证	1.4.3
2.3.1. 客户端密码	1.4.3.1
2.3.2. 其他客户端认证方式	1.4.3.2
2.4. 未注册客户端	1.4.4
3. 协议端点	1.5
3.1. 授权端点	1.5.1
3.1.1. 响应类型	1.5.1.1
3.1.2. 重定向端点	1.5.1.2
3.1.2.1. 端点请求私密性	1.5.1.2.1
3.1.2.2. 需要注册	1.5.1.2.2
3.1.2.3. 动态配置	1.5.1.2.3
3.1.2.4. 无效端点	1.5.1.2.4
3.1.2.5. 端点内容	1.5.1.2.5
3.2. 令牌端点	1.5.2
3.2.1. 客户端认证	1.5.2.1
3.3. 访问令牌作用域	1.5.3
4. 获取授权 (机器翻译)	1.6
5. 签发 Access Token (待翻译)	1.7

6. 刷新 Access Token (待翻译)	1.8
7. 访问受保护的资源 (待翻译)	1.9
8. 扩展性 (待翻译)	1.10
9. 原生App (待翻译)	1.11
10. 安全考虑 (待翻译)	1.12
11. IANA Considerations (待翻译)	1.13
12. 参考 (待翻译)	1.14
附录 A (待翻译)	1.15
附录 B (待翻译)	1.16
附录 C (待翻译)	1.17

OAuth 2.0 授权框架

RFC6749 中文版

qunfanyi.com 翻译

《OAuth2 授权框架标准[RFC6749]》

!!!!完成度 20%

下载

- PDF 格式
 - EPUB 格式
 - MOBI 格式
-

欢迎加入 qunfanyi.com 一块翻译 IT 技术文档，为中国 IT 贡献力量

英文原文地址：<https://tools.ietf.org/html/rfc6749>

翻译者：老唐35 2018/8 QQ : 840750575 email: 840750575@qq.com github: <https://github.com/qunfanyi/rfc6749>

请不要用于商业目的

OAuth 2.0 授权框架

RFC6749 中文版

qunfanyi.com 翻译

《OAuth2 授权框架标准[RFC6749]》

!!!!完成度 10%

欢迎加入 qunfanyi.com 一块翻译 IT 技术文档，为中国 IT 贡贡献力量

英文原文地址: <https://tools.ietf.org/html/rfc6749>

翻译者: 老唐35 2018/8 QQ : 840750575 email: 840750575@qq.com github: <https://github.com/qunfanyi>

请不要用于商业目的

1. 介绍

在简单的客户端——服务端授权模型中，客户端使用服务器发给资源归属者的凭据（密码、证书之类）请求受保护的资源。第三方应用访问受保护资源时，需要资源归属者向第三方提供凭据。这造成了一些问题和限制：

- 第三方应用需要存储资源归属者的凭据以备将来访问，典型的如未加密的密码。
- 服务器需要支持密码授权，密码方式有安全弱点。
- 资源归属者无法限制第三方应用只允许访问部分受保护资源。
- 资源归属者不能单独撤销个别访问权限，除非改掉所有第三方的密码。
- 任何第三方应用程序的妥协都会导致最终用户的密码和所有受该密码保护的数据受到损害。

OAuth 为资源归属者引入角色及授权层解决这些问题。OAuth 客户端在资源所有者及资源服务器控制下访问资源，且使用与资源所有者不一样的凭据。客户端不再使用资源所有者凭据去访问资源，而是使用访问令牌（一串表示范围、期限及其他访问属性的特殊文字）去访问。Access Token (访问令牌) 是资源所有人同意后才由授权服务器颁发给第三方客户端。客户端使用 Access Token 访问资源服务器上的受保护资源。

例如：一个用户（资源所有人）可以授权给打印服务商（客户端）访问她存储在照片服务商（资源服务器）的照片（资源），不需要把她的账号密码告诉打印服务商。她使用照片服务商信任的服务器 同意照片服务商（授权服务器）授权给打印服务商(客户端)特定的访问凭据 (Access Token)。

这份标准为 HTTP([RFC2616]) 设计，但是也可以在其它非 HTTP 协议上使用。之前 OAuth 1.0 协议([RFC5849])，在社区努力下已经正式发布。这份标准吸收 OAuth 1.0 了精华，从 IETF 社区扩展了一些需求。OAuth 2.0 协议不向后兼容 OAuth 1.0。两个版本会并行，有些可能会两个版本都支持。但是，建议实现 OAuth 2.0 标准，OAuth 1.0 只用于遗留系统。OAuth 2.0 与 OAuth 1.0 共通之处很少。实现 OAuth 2.0 不需要了解 OAuth 1.0。

1.1. 角色

OAuth 定义了4种角色：

- 资源所有者
能对请求访问授权的实体。如果是个人话，可以理解为终端用户。
 - 资源服务器
存放受保护的资源，能接收响应携带访问令牌的请求操作。
 - 客户端
资源所有者授权后对受保护资源发出操作请求的应用程序。
术语“客户端”跟实现方式无关（运行在服务器、桌面或其他设备都可以）
 - 授权服务器
在获得资源所有者成功授权后给客户端颁发访问令牌。
-

授权服务器与资源服务器之间的交互，在本文后边。授权服务器跟资源服务器可以是同一服务器也可以是分开的。单个授权服务器可以签发多个资源服务器接受令牌。

1.2. 协议流程

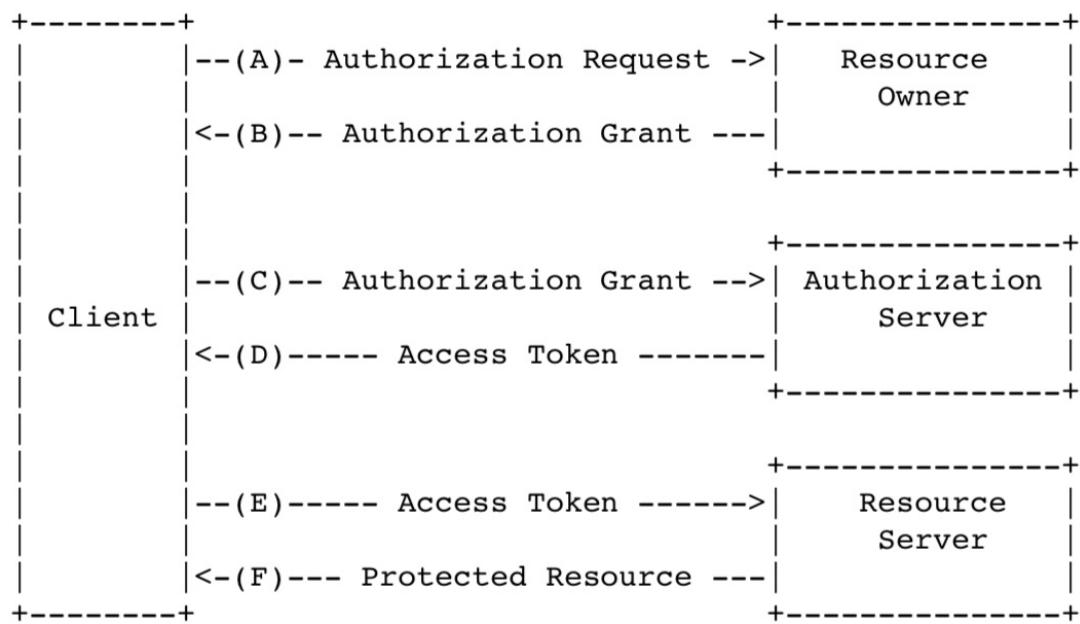


图1 抽象协议流程

图1中的抽象 OAuth 2.0 流程 描述了4种角色之间的交互，包含以下步骤：

(A) 客户端向资源所有者请求授权。

可以直接请求资源所有者授权，更好的方式是通过授权服务器中转间接授权。

(B) 客户端接收到授权许可（表示资源所有者授权的凭据）是用本标准定义的四种许可模式之一或扩展的许可模式。

授权许可模式依赖客户端请求的授权类型及授权服务器支持情况。

(C) 客户端使用授权许可请求授权服务器颁发访问令牌。

(D) 授权服务器验证客户端及授权许可，有效则颁发访问令牌。

(E) 客户端使用访问令牌访问资源服务器上受保护的资源。

(F) 资源服务器验证访问令牌，有效则处理访问请求。

客户端从资源所有者获取授权许可时（步骤(A) 及 (B)中所述）更好的方式是通过授权服务器中转如4.1章 图 3所述。

1.3. 授权许可

授权许可表示资源所有者授权的凭据（授权访问她的受保护资源），用于客户端获取访问令牌。本标准定义了四种许可模式——授权码，隐式授权，资源所有者密码凭据及客户端凭据。此外也可以通过扩展机制定义其他许可模式。

1.3.1. 授权码

授权码是客户端通过授权服务器从资源所有者间接获得。

不像直接从资源所有者授权，客户端引导资源所有者到授权服务器（通过 [RFC2616] 定义的 user-agent），然后再带着授权码返回客户端。

再带着授权码返回客户端之前，授权服务器需要资源所有者授权。

由于资源所有者只需要向授权服务器授权，所以资源所有者的凭据不会被客户端知晓。

授权代码提供了一些重要的安全好处，例如对客户端进行身份验证的能力，以及直接将访问令牌传输到客户端，而不通过资源所有者的用户代理传递访问令牌，并可能将其暴露给其他人(包括资源所有者)。

1.3.2. 隐式授权

隐式授权是一种授权码简化流程为浏览器中用脚本语言（如javascript）实现的客户端而优化。
在隐式授权流程中，资源所有者授权后不再颁发授权码给客户端，而是直接颁发访问令牌给客户端。
隐式授权没有颁发中间凭据（如用于获取访问令牌的授权码）。
在隐式授予流期间发出访问令牌时，授权服务器不对客户端进行身份验证。
有时候能通过携带访问令牌的重定向网址验证客户端身份。
访问令牌可能会暴露给资源所有者或其他能访问资源所有者 user-agent 的应用。
隐式授权由于减少了获取访问令牌的开销，提升了一些客户端响应性及效率（如浏览器中的客户端）。
然而，隐式授权的方便与安全性需要权衡下，可以参考16节，特别是可以使用授权码的情况下。

1.3.3. 资源所有者密码凭据

资源所有者密码凭据(即用户名和密码)可直接用作授权许可，以获得访问令牌。只有在资源所有者和客户端之间存在高度信任时(如，客户端是设备操作系统的一部分或高权限的应用)，且当其他授权许可模式(如授权代码)不可用时采用凭据许可模式。即使此授予类型要求客户直接访问资源所有者凭据，资源所有者凭据也用于单个请求并交换为访问令牌。客户端可以用凭据交换长期有效的访问令牌或刷新令牌，不需要存储资源所有者凭据。

1.3.4. 客户端凭据

当授权范围是在客户端控制范围内或者客户端与授权服务器协同保护资源时，可以使用客户端凭据许可模式。

使用客户端凭据许可模式典型的场景是当客户端访问自己的资源（客户端同时也是资源所有者）或者根据之前授权服务器安排正在访问受保护资源。

1.4. 访问令牌

访问令牌是访问受保护资源的凭据。访问令牌是发给客户端的一串字符串表示授权可以访问。客户端一般无需理解这串字符。令牌由资源所有者指定的访问范围及期限，由资源服务器及授权服务器遵照执行。令牌可能是一个用于获取授权信息的标示符，或者令牌中直接包含授权信息（如，由数据及签名组成的令牌字符串）。为了使客户端使用令牌，可能需要超出本规范范围的其他身份验证凭据。访问令牌提供抽象层，用资源服务器理解的单个令牌替换不同的授权结构（例如用户名和密码）。这种抽象颁发访问令牌比获取令牌有更多限制，资源服务器也无需理解具体的授权方式。访问令牌可以为了满足资源服务器的不同加密方式，有不同的格式、构造、实用方法（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，并由配套规范（如[RFC6750]）定义。

1.5. 刷新令牌

刷新令牌用于重新获取访问令牌。当前访问令牌要过期或失效之前，刷新令牌使用授权服务器颁发给客户端的刷新令牌获取新的访问令牌。或者使用刷新令牌获取新的访问令牌，新访问令牌访问范围不能大于当前令牌的范围（新访问令牌可能有效期更短或权限更少）。授权服务器自行决定是否颁发刷新令牌。如果授权服务器颁发刷新令牌，需要与访问令牌同时颁发（见图1步骤（D））。刷新令牌是一串字符，表示资源所有者给客户端的授权。这串字符客户端一般不需要理解。令牌是获取授权信息的标示符。不像访问令牌，刷新令牌只用于授权服务器，不会发给资源服务器。

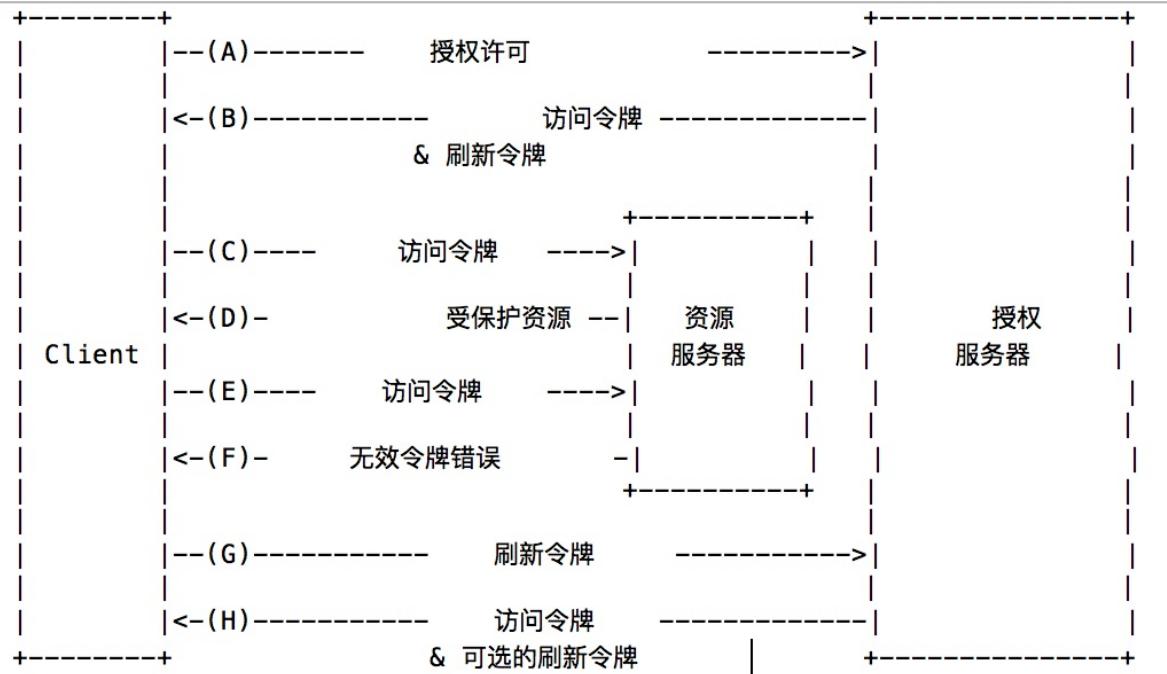


图2: 刷新过期的访问令牌 图2所示流程包括下列步骤: (A) 客户端用授权服务器颁发的授权许可获取访问令牌。 (B) 授权服务器认证客户端验证授权许可，有效的话颁发访问令牌及刷新令牌。 (C) 客户端用访问令牌访问保护资源。 (D) 资源服务器验证访问令牌，有效的话处理访问请求。 (E) 步骤 (C) 及 (D) 循环直到访问令牌过期。如果客户端知道访问令牌过期，跳到步骤(G); 否则可继续访问保护资源。 (F) 访问令牌无效后，资源服务器返回“无效令牌错误”。 (G) 客户端使用刷新令牌向授权服务器请求访问令牌。客户端身份验证要求基于客户端类型和授权服务器策略。 (H) 授权服务器认证客户端并验证刷新令牌，如果有效则颁发新的访问令牌（同时可选颁发新的刷新令牌） 步骤(C)、(D)、(E) 及 (F) 如第7章所述不在本标准范围内。

1.6. TLS 版本

每当此规范使用传输层安全性(TLS)时，根据广泛部署和已知的安全漏洞，TLS的适当版本(或多个版本)将随着时间的推移而不同。

在撰写本文时，TLS 1.2 版[RFC 5246]是最新版本，但部署基础非常有限，可能难以实现。TLS 1.0版 [RFC2246]是最广泛的部署版本，会提供最广泛的互操作性。实现时还可以支持别的传输层安全机制，以满足附加的安全性要求的。

1.7. Http 重定向

该规范广泛使用HTTP重定向，其中包括 客户端或授权服务器将资源所有者的 user-agent(译者：浏览器、应用) 引导到另一个目的地。 (译者：重定向到别的网址) 虽然本规范中的示例使用HTTP 302状态代码，但可以通过其他user-agent重定向的方法，并被视为实现细节。 (译者：可以用别的方法重定向)

1.8. 互操作

OAuth 2.0 提供各式各样安全属性定义良好的授权框架。但是，作为一个包含许多可选组件的丰富且高度可扩展的框架，该规范本身很可能会产生范围广泛的不可互操作的实现。此外，该规范还保留了一些部分或完全未定义的所需组件(例如，客户端注册、授权服务器功能、端点发现)。缺乏这些组件，为了实现交互客户端必须为特定的授权服务器及资源服务器手动特殊配置。该框架的设计带有明确的期望，即未来的工作将定义必要的规范配置文件和扩展，以实现完全的web规模互操作性。

1.9. 符号约定

本标准中的关键词“必须”、“禁止”、“需要”、“将”、“将不”、“可能”、“可能不”、“建议”、“或许”及“可选”在[RFC2119]中解释。本标准使用 ABNF [RFC5234] 另外 URI引用 请参考 RFC3986 。下列安全相关的术语参考 RFC4949，包括但不限于：攻击、认证、授权、证书、保密协议、凭据、加密、标示、签名、信任、验证及校验 除非另有说明，所有协议参数及取值都是大小写敏感的。

在协议初始化之前，客户端向授权服务器注册。

客户端向授权服务器注册的手段超出了本规范的范围，但通常涉及与HTML注册表单的最终用户交互。

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，注册可以依靠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，客户端注册可以使用自发布或第三方签发，或者由授权服务器使用可信通道自动发现客户端。

当注册客户端时，客户开发人员应：

- 指定客户类型，如第2.1节所述
- 提供客户重定向URI，第3.1.2节所述的
- 包括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、法律条款的接受)。

2.1. 客户端类型

OAuth基于客户端与授权服务器安全地进行身份验证的能力（即，保证客户端凭证安全的能力）定义了两种客户端类型：

- 私密客户端（confidential）

一种是能够维护其凭证的机密性的机密客户端（例如，在限制访问的私密服务器上实现的客户端）或使用其他手段保证安全的客户端。

- 公共客户端（public）

另一种是公共客户端不能维护其凭证的机密性（例如，在资源所有者使用的设备上执行的客户端，例如安装的本地应用程序或基于web浏览器的应用程序），并且不能通过任何其他手段保护客户端身份验证安全。

客户端类型设计基于授权服务器的安全身份验证定义及可承受客户端凭证暴露程度。

授权服务器不应假设客户端类型。

客户端可能由一些分布式组件实现，每个组件具有不同的客户端类型和安全上下文（例如，由一些机密服务器的组件和公共浏览器组件组成的分布式客户端）。

如果授权服务器不为这些客户端提供支持或者不提供关于其注册的指导，则客户端应该将每个组件注册为单独的客户端。

本规范围绕以下客户端设计的：

- Web应用程序（web application）Web应用程序是运行在Web服务器上的机密客户端。客户端凭证以及发给客户端的任何访问令牌都存储在Web服务器上，并且不向资源所有者公开或访问。

- 基于用户代理的应用程序是公共客户端（user-agent-based application）其中客户端代码从web服务器下载并运行在资源所有者设备（译者：如电脑、手机）的用户代理中（例如，web浏览器）。

协议数据和凭据很容易访问（一般能看见）给资源所有者。

由于此类应用程序驻留在用户代理中，因此它们在请求授权时可以无缝地使用用户代理功能。

- 原生应用（native application）原生应用是在资源所有者设备上安装并执行的公共客户端。资源所有者可以访问协议数据和凭据。假设客户端可以提取包含在应用程序中的任何客户端认证凭据。另一方面，动态签发的凭据（如访问令牌或刷新令牌）需要提供可接受的保护级别。至少，这些凭据受到保护，以免受相关恶意服务器的影响。这些凭据还需要避免被驻留在同一设备上的其他应用程序访问。

2.2. 客户端标识

授权服务器向注册的客户端发出一个客户端标识符——表示客户端提供的注册信息的唯一字符串。

客户端标识符不是私密的；它暴露于资源所有者，不能单独用于客户端身份验证。

客户端标识符对于授权服务器都是唯一的。

客户端标识符字符串大小本规范未定义。

客户端应该避免对标识符大小做出假设。

授权服务器应该记录它所发出的任何标识符的大小。

2.3. 客户端认证

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于授权服务器的安全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认证。

机密客户端通常发布（或建立）一组客户端凭据，用于与授权服务器进行身份验证（例如，密码、公钥/私钥对）。

授权服务器可以建立与公共客户端的客户端认证方法。

但是，授权服务器不能依赖公共客户端身份验证来识别客户端。

客户端不能在一个请求中使用多种身份验证方法。

2.3.1. 客户端密码

拥有客户端密码的客户端可以使用[RFC2617]中定义的HTTP基本身份验证方案来与授权服务器进行身份验证。

客户端标识符使用每个附录B的“application/x-www-form-urlencoded”编码算法进行编码，并且编码值用作用户名；客户端密码使用相同的算法进行编码并且用作密码。

授权服务器必须支持HTTP Basic身份验证方案，用于签发过密码的客户端进行身份验证。

或者，授权服务器也可支持在请求主体中使用参数表示客户端凭据：

- `client_id` 必须。向客户端签发的客户端标识符在第2.2节注册过程中描述。
- `client_secret`
必填。客户端密码。如果客户端秘密是空字符串，则客户端可以省略参数。

不推荐使用请求主体中使用两个参数表示凭据的方式，应该直接利用HTTP Basic 授权机制（或者其他基于密码的 http 授权机制）

这些参数只能在请求体中传输，不能包含在请求URI中。

例如，使用主体参数刷新访问令牌的请求（第6节）：

```
POST /token
HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
grant_type=refresh_token&refresh_token=tGzv3J0kF0XG5Qx2T1KWIA &
client_id=s6BhdRkqt3&client_secret=7FjfpoZBr1KtDRbnfVdmIw
```

授权服务器必须要求使用 TLS 发送密码认证请求（参考1.6）

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端点免受暴力攻击。

2.3.2. 其他客户端认证方式

授权服务器可以支持与其安全性要求相匹配的任何合适的HTTP认证方案。

当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。

2.4. 未注册客户端

本规范不排除未注册客户端的使用。

然而，这种客户端的使用超出了本规范的范围，需要额外的安全性分析和对其互操作性影响的审查。

3. 协议端点

授权过程使用两个授权服务器端点（HTTP资源）：

- 授权端点
客户端使用该端点通过用户-代理重定向从资源所有者获得授权。
- 令牌端点 用于客户端将授权许可交换成访问令牌，通常需要客户端身份验证。
- 以及一个客户端端点： 授权服务器使用该端点通过资源所有者用户代理向客户端返回包含授权凭证的响应。

并非每个授权类型都使用所有端点。

扩展授权类型可以根据需要定义附加端点。

3.1. 授权端点

授权过程使用两个授权服务器端点（HTTP资源）：

- 授权端点——客户端使用该端点通过用户-代理重定向从资源所有者获得授权许可。
- 令牌端点-客户端把授权许可换成访问令牌，通常使用客户端身份验证。

以及一个客户端端点：

- Redirection端点——授权服务器使用该端点通过资源所有者用户代理向客户端返回包含授权凭证的响应。

并非每个授权授予类型都使用端点。

扩展授权类型可以根据需要定义附加端点。

3.1.1. 响应类型

授权端点由授权码授权类型和隐式授权类型流程使用。

客户端使用以下参数向授权服务器告知所需的授予类型：所需的响应类型。

该值必须是用于请求授权代码的“代码”之一，如第4.1.1节所述的“令牌”，用于请求如4.2.1节所描述的访问令牌（隐式授予），或如第8.4节所述的注册扩展值。

扩展响应类型可能包含以空格分隔（%x20）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是由它们各自的规范来定义的。

如果授权请求缺少“response_type”参数，或者如果不理解响应类型，授权服务器必须返回错误响应，如4.1.2.1节所述。

3.1.2. 重定向端点

在完成与资源所有者的交互之后，授权服务器将资源所有者的用户代理引导回客户端。

授权服务器将用户代理重定向到客户端重定向端点，客户端重定向端点是以前在注册客户端时注册的，或在授权请求时传过去的。

重定向端点URI必须是由[RFC3986]第4.3节定义的绝对URI。

端点URI可能包括“application/x-www-form-urlencoded”（每个附录B）格式化的查询组件（[RFC3986]部分3.4），在添加其他查询参数时必须保留该组件。

端点URI不能包含片段组件。（译者：不能包含#部分）

3.1.2.1. 端点请求私密性

当请求的响应类型是“代码”或“令牌”时，或者当重定向请求将导致敏感凭据在开放网络上传输时，重定向端点应该要求使用第1.6节中所描述的TLS。

该规范并不强制使用TLS，因为在撰写本文时，要求客户机部署TLS是许多客户机开发人员的一个重大障碍。

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不安全的端点（例如，在授权请求期间显示消息）。

传输层安全性的缺乏可能对客户端及其授权访问的受保护资源的安全性产生严重影响。

当授权过程被用作委托端用户验证的形式（例如，第三方登录服务）时，传输层安全性的使用尤为关键。

3.1.2.2. 需要注册

授权服务器必须要求下列客户端注册它们的重定向端点：

- 公共客户端。
- 使用隐式授予类型的机密客户。

授权服务器应该要求所有客户端在使用授权端点之前注册它们的重定向端点。

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”请求参数来实现每个请求的定制）。

如果不可能要求注册完整的重定向URI，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地改变重定向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点。

缺少重定向URI注册要求可使攻击者能够使用授权端点作为开放重定向器，如第10.15节所述。

3.1.2.3. 动态配置

如果已经注册了多个重定向URI，如果仅注册了部分重定向URI，或者如果没有注册重定向URI，则客户端必须使用“redirect_uri”请求参数在授权请求中包括重定向URI。

当在授权请求中包括重定向URI时，如果注册了任何重定向URI，则授权服务器必须将接收到的值与[RFC3986]第6节中定义的至少一个已注册重定向URI（或URI组件）进行比较和匹配。

如果客户端注册包括完整的重定向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来比较这两个URI。

3.1.2.4. 无效端点

如果授权请求由于缺少、无效或不匹配的重定向URI而验证失败，则授权服务器应将错误通知资源所有者，不要自动将用户代理重定向到无效重定向URI。

3.1.2.5. 端点内容

对客户端端点的重定向请求通常是由用户代理处理的HTML文档响应（译者：http 3xx 之类）。

如果作为重定向请求的结果直接提供HTML响应，则HTML文档中包含的任何脚本都将能完全访问重定向URI及其包含的凭据。

客户端不应该在重定向端点响应中包括任何第三方脚本（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，而不公开凭据（在URI或其他地方）。

如果包括第三方脚本，则客户端必须确保其自己的脚本（用于从URI中提取和删除凭据）首先执行。

3.2. 令牌端点

令牌端点用于客户端拿授权许可或刷新令牌换取访问令牌。

除了隐式授予类型（因为直接发出访问令牌），其他授权类型都需要使用令牌端点。

这意味着虽然获取令牌端点网址的方式超出了本规范范围，但令牌端点网址一般在授权服务文档中有。

端点网址可能包括“application/x-www-form-urlencoded”（每个附录B）格式化的查询组件（[RFC3986]部分3.4），在添加其他查询参数时必须保留该组件。

端点URI不能包含片段(译者：网址不能包含#部分)。

由于传输令牌端点请求及返回结果都是明文凭据（在HTTP请求和响应中），授权服务器在向令牌端点发送请求时必须要求使用第1.6节中所描述的TLS。

客户端在访问令牌请求时必须使用HTTP“POST”方法。

无值发送的参数必须被处理，就像它们从请求中被省略一样。

授权服务器必须忽略未识别的请求参数。

请求和响应参数不能多次出现。

3.2.1. 客户端认证

当向令牌端点发出请求时，机密客户机或其他发出客户机凭证的客户机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代码绑定到发出给它们的客户端。

当授权代码通过不安全的通道传输到重定向端点时，或者当重定向URI没有完全注册时，客户端身份验证是关键的。

- 通过禁用客户端或更改其凭证从受损客户端恢复，从而防止攻击者滥用被窃取的刷新令牌。

更改单个客户端证书集比撤销整个刷新令牌组要快得多。

- 实现认证管理最佳实践，这需要定期凭证旋转。

整个刷新令牌集合的旋转是具有挑战性的，而单组客户端证书的旋转更容易。

客户端可能在向令牌端点发送请求时使用“client_id”请求参数来标识自己。

在对令牌端点的“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。

这保护客户端不必替换认证代码。

3.3. 访问令牌作用域

授权和令牌端点允许客户端使用“scope”参数指定访问请求的范围。

反过来，授权服务器使用“scope”响应参数向客户端通知发出的访问令牌的范围。

scope 参数的值使用空格分隔的、区分大小写的字符串列表表示。

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，则它们的顺序并不重要，并且每个字符串都向所请求的范围添加了额外的访问范围。

授权服务器可以根据授权服务器策略或资源所有者的指令完全或部分地忽略客户端请求的范围。

如果签发的访问令牌范围与客户端请求的访问令牌范围不同，则授权服务器必须包括“scope”响应参数，以向客户端通知所授予的实际范围。

如果客户端在请求授权时省略了范围参数，则授权服务器必须使用预定义的默认值处理请求，或者使指示无效范围的请求失败。

授权服务器文档中应记录其范围要求和默认值（如果已定义）。

4. 获取授权许可

为了请求访问令牌，客户端从资源所有者获得授权。

授权以授权许可的形式表示，客户端使用它来请求访问令牌。

OAuth 定义了四种授权许可类型：授权码、隐式、资源所有者密码凭据和客户端凭据。

它还提供了用于定义附加授权许可类型的扩展机制。

5. Issuing an Access Token

[en]If the access token request is valid and authorized, the authorization server issues an access token and optional refresh token as described in Section 5.1.

[zh_CN]如果访问令牌请求是有效的并被授权的，授权服务器将发布访问令牌和可选的刷新令牌，如5.1节所述。

[en]If the request failed client authentication or is invalid, the authorization server returns an error response as described in Section 5.2.

[zh_CN]如果请求客户端身份验证失败或无效，授权服务器将返回第5.2节中描述的错误响应。

6. Refreshing an Access Token

[en]If the authorization server issued a refresh token to the client, the client makes a refresh request to the token endpoint by adding the following parameters using the "application/x-www-form-urlencoded" format per Appendix B with a character encoding of UTF-8 in the HTTP request entity-body: grant_type REQUIRED.

[zh_CN]如果授权服务器向客户端发出了刷新令牌，则客户端通过使用HTTP请求实体中的UTF-8字符编码的“application/x-www-form-urlencoded”格式为每个附录B添加以下参数，向令牌端点发出刷新请求：grant需要的类型。

[en]Value MUST be set to "refresh_token".

[zh_CN]值必须设置为“刷新符号”。

[en]refresh_token REQUIRED.

[zh_CN]需要刷新标记。

[en]The refresh token issued to the client.

[zh_CN]向客户端发出的刷新令牌。

[en]scope OPTIONAL.

[zh_CN]可选范围。

[en]The scope of the access request as described by Section 3.3.

[zh_CN]如第3.3节所述的访问请求的范围。

[en]The requested scope MUST NOT include any scope not originally granted by the resource owner, and if omitted is treated as equal to the scope originally granted by the resource owner.

[zh_CN]所请求的范围必须不包括最初未由资源所有者授予的任何范围，如果省略，则被视为与资源所有者最初授予的范围相同。

[en]Because refresh tokens are typically long-lasting credentials used to request additional access tokens, the refresh token is bound to the client to which it was issued.

[zh_CN]因为刷新令牌通常是用于请求附加访问令牌的持久凭证，所以刷新令牌绑定到向其发布的客户端。

[en]If the client type is confidential or the client was issued client credentials (or assigned other authentication requirements), the client MUST authenticate with the authorization server as described in Section 3.2.1.

[zh_CN]如果客户机类型是保密的，或者客户机被颁发了客户机凭证（或者被指派了其他身份验证要求），则客户机必须如3.2.1节所述，使用授权服务器进行身份验证。

[en]For example, the client makes the following HTTP request using transport-layer security (with extra line breaks for display purposes only): 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=tGzv3JOkF0XG5Qx2TIKWIA Hardt Standards Track [Page 47] RFC 6749 OAuth 2.0 October 2012 The authorization server MUST: o require client authentication for confidential clients or for any client that was issued client credentials (or with other authentication requirements), o authenticate the client if client authentication is included and ensure that the refresh token was issued to the authenticated client, and o validate the refresh token.

[zh_CN]例如，客户机使用传输层安全性发出以下HTTP请求（仅为了显示目的使用额外的换行）：POST/令牌
HTTP/1.1 Host: server.example.com 授权: Basic czZCaGRSa3F0MzpnWDFmF0M2JW 内容类型: application/x-www-form-urlencoded grant_type=esh_token&esh_token=tGzv3JOkF0XG5Qx2TIKWIA 硬标准跟踪[第47页]RFC 6749
OAuth 2.0 2012年10月2日授权服务器必须：
o 要求对机密客户机或对颁发客户机凭据（或其他身份验证要求）的任何客户机进行客户机身份验证，
o 如果包括客户端身份验证，则对客户端进行身份验证，并确保将刷新令牌颁发给经过身份验证的客户端，并且验证刷新令牌。

[en]If valid and authorized, the authorization server issues an access token as described in Section 5.1.

[zh_CN]如果有效和授权，授权服务器发出访问令牌，如第5.1节所述。

[en]If the request failed verification or is invalid, the authorization server returns an error response as described in Section 5.2.

[zh_CN]如果请求失败验证或无效，则授权服务器返回错误响应，如第5.2节所述。

[en]The authorization server MAY issue a new refresh token, in which case the client MUST discard the old refresh token and replace it with the new refresh token.

[zh_CN]授权服务器可以发出新的刷新令牌，在这种情况下，客户端必须丢弃旧的刷新令牌并用新的刷新令牌替换它。

[en]The authorization server MAY revoke the old refresh token after issuing a new refresh token to the client.

[zh_CN]授权服务器可以在向客户端发布新刷新令牌之后撤销旧刷新令牌。

[en]If a new refresh token is issued, the refresh token scope MUST be identical to that of the refresh token included by the client in the request.

[zh_CN]如果发出新的刷新令牌，刷新令牌范围必须与请求中包含的客户端所包含的刷新令牌相同。

7. Accessing Protected Resources

[en]The client accesses protected resources by presenting the access token to the resource server.

[zh_CN]客户端通过将访问令牌呈现给资源服务器来访问受保护资源。

[en]The resource server MUST validate the access token and ensure that it has not expired and that its scope covers the requested resource.

[zh_CN]资源服务器必须验证访问令牌，并确保它没有过期，并且其范围覆盖所请求的资源。

[en]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.

[zh_CN]资源服务器用于验证访问令牌（以及任何错误响应）的方法超出了本规范的范围，但通常涉及资源服务器和授权服务器之间的交互或协调。

[en]The method in which the client utilizes the access token to authenticate with the resource server depends on the type of access token issued by the authorization server.

[zh_CN]客户端利用访问令牌与资源服务器进行身份验证的方法取决于授权服务器发出的访问令牌的类型。

[en]Typically, it involves using the HTTP "Authorization" request header field [RFC2617] with an authentication scheme defined by the specification of the access token type used, such as [RFC6750].

[zh_CN]通常，它涉及使用HTTP“Authorization”请求报头字段[RFC2617]和由所使用的访问令牌类型的规范定义的身份验证方案，例如[RFC6750]。

8. Extensibility

9. Native Applications

[en]Native applications are clients installed and executed on the device used by the resource owner (i.e., desktop application, native mobile application).

[zh_CN]本机应用程序是在资源所有者（即，桌面应用程序、本机移动应用程序）使用的设备上安装和执行的客户机。

[en]Native applications require special consideration related to security, platform capabilities, and overall end-user experience.

[zh_CN]本地应用需要与安全性、平台能力和整体最终用户体验相关的特殊考虑。

[en]The authorization endpoint requires interaction between the client and the resource owner's user-agent.

[zh_CN]授权终结点需要客户端和资源所有者的用户代理之间的交互。

[en]Native applications can invoke an external user-agent or embed a user-agent within the application.

[zh_CN]本地应用程序可以调用外部用户代理或在应用程序中嵌入用户代理。

[en]For example:

- o External user-agent - the native application can capture the response from the authorization server using a redirection URI with a scheme registered with the operating system to invoke the client as the handler, manual copy-and-paste of the credentials, running a local web server, installing a user-agent extension, or by providing a redirection URI identifying a server-hosted resource under the client's control, which in turn makes the response available to the native application.

[zh_CN]例如：外部用户代理-本机应用程序可以使用重定向URI从授权服务器捕获响应，该URI具有与操作系统注册的方案，以调用客户端作为处理器、手动复制和粘贴证书，运行本地Web服务器，拖延用户代理扩展，或者通过在客户端的控制下提供标识服务器托管资源的重定向URI，这反过来使响应对本机应用可用。

[en]o Embedded user-agent - the native application obtains the response by directly communicating with the embedded user-agent by monitoring state changes emitted during the resource load, or accessing the user-agent's cookies storage.

[zh_CN]o嵌入式用户代理——本地应用程序通过监视在资源加载期间发出的状态变化或者访问用户代理的cookie存储来与嵌入式用户代理直接通信来获得响应。

[en]When choosing between an external or embedded user-agent, developers should consider the following:

- o An external user-agent may improve completion rate, as the resource owner may already have an active session with the authorization server, removing the need to re-authenticate.

[zh_CN]当在外部用户代理或嵌入式用户代理之间进行选择时，开发人员应该考虑以下几点：

- o外部用户代理可以提高完成率，因为资源所有者可能已经与授权服务器有一个活动会话，因此不需要重新身份验证。

[en]It provides a familiar end-user experience and functionality.

[zh_CN]它提供了一个熟悉的终端用户体验和功能。

[en]The Hardt Standards Track [Page 52] RFC 6749 OAuth 2.0 October 2012 resource owner may also rely on user-agent features or extensions to assist with authentication (e.g., password manager, 2-factor device reader).

[zh_CN]硬标准跟踪[第52页]RFC 6749 OAuth 2.0 2012年10月2.0资源所有者还可以依靠用户代理特性或扩展来辅助身份验证（例如，密码管理器、2因素设备读取器）。

[en]o An embedded user-agent may offer improved usability, as it removes the need to switch context and open new windows.

[zh_CN]嵌入式用户代理可以提供改进的可用性，因为它消除了切换上下文和打开新窗口的需要。

[en]o An embedded user-agent poses a security challenge because resource owners are authenticating in an unidentified window without access to the visual protections found in most external user-agents.

[zh_CN]o嵌入式用户代理提出了一个安全挑战，因为资源所有者在未识别的窗口中进行身份验证，而没有访问大多数外部用户代理中找到的视觉保护。

[en]An embedded user-agent educates end-users to trust unidentified requests for authentication (making phishing attacks easier to execute).

[zh_CN]嵌入式用户代理教育终端用户信任身份验证的未标识请求（使得钓鱼攻击更容易执行）。

[en]When choosing between the implicit grant type and the authorization code grant type, the following should be considered:
o Native applications that use the authorization code grant type SHOULD do so without using client credentials, due to the native application's inability to keep client credentials confidential.

[zh_CN]当在隐式授权类型和授权代码授权类型之间进行选择时，应该考虑以下因素：
o 使用授权代码授权类型的本地应用程序应该不使用客户端凭证，因为本地应用程序不能保留客户端凭证LS机密。

[en]o When using the implicit grant type flow, a refresh token is not returned, which requires repeating the authorization process once the access token expires.

[zh_CN]o 当使用隐式授予类型流时，不返回刷新令牌，这需要在访问令牌过期后重复授权过程。

10. Security Considerations

[en]As a flexible and extensible framework, OAuth's security considerations depend on many factors.

[zh_CN]作为一个灵活的和可扩展的框架，OAuths的安全考虑取决于许多因素。

[en]The following sections provide implementers with security guidelines focused on the three client profiles described in Section 2.1: web application, user-agent-based application, and native application.

[zh_CN]以下各节向实现者提供了安全指南，这些指南集中于第2.1节中描述的三个客户机概要文件：web应用程序、基于用户代理的应用程序和本地应用程序。

[en]A comprehensive OAuth security model and analysis, as well as background for the protocol design, is provided by [OAuth-THREATMODEL].

[zh_CN]一个全面的OAUTH安全模型和分析，以及协议设计的背景，由[OAuth-ThreatModel]提供。

11. IANA Considerations

12. References

a

b

C