Table of Contents

RFC6749.zh-cn

A translation in Simplified Chinese for RFC 6749-The OAuth 2.0 Authorization Framework.

RFC 6749-OAuth 2.0授权框架简体中文翻译。

TODO

总体翻译已完成，希望大家帮忙完善下哦。

开始阅读

1. 简介

在传统的客户端-服务器身份验证模式中，客户端请求服务器上访问受限的资源（受保护的资源）时，需要使用资源所有者的凭据在服务器上进行身份验证。 资源所有者为了给第三方应用提供受限资源的访问权限，需要与第三方共享它的凭据。这就导致一些问题和局限：

- 第三方应用需要存储资源所有者的凭据以供将来使用。该凭据通常是明文密码。
- 服务器需要支持密码身份认证，尽管密码认证有固有的安全缺陷。
- 第三方应用获得了对资源所有者的受保护资源的过于宽泛的访问权限，从而导致资源所有者不能限制对资源的有限子集的访问时限或权限。
- 资源所有者不能撤销某个第三方的访问权限而不影响其它第三方，并且必须更改他们的密码才能做到。
- 与任何第三方应用的妥协导致对终端用户的密码及该密码所保护的所有数据的妥协。

OAuth通过引入授权层以及从资源所有者角色分离出客户端角色来解决这些问题。在OAuth中，客户端请求对受资源所有者控制且托管在资源服务器上的资源的访问权限，并授予一组不同于资源所有者所拥有的凭据。

作为使用资源所有者的凭据访问受保护资源的替代，客户端获得一个访问令牌———一个代表特定作用域、生命周期以及其他访问权限属性的字符串。访问令牌由授权服务器在资源所有者认可的情况下颁发给第三方客户端。客户端使用访问令牌访问托管在资源服务器上的受保护资源。

例如，终端用户（资源所有者）可以许可一个打印服务（客户端）访问她存储在图片分享网站（资源服务器）上的受保护图片，而无需与打印服务分享自己的用户名和密码，而是，她直接与图片分享网站信任的服务器（授权服务器）进行身份验证，该服务器颁发给打印服务具体的委托凭据（访问令牌）。

本规范是为HTTP（RFC2616）协议设计的。在任何非HTTP协议上使用OAuth不在本规范的范围之内。

OAuth 1.0协议（RFC5849）作为一个指导性文档发布，是一个小的特设团体的工作成果。本标准化规范在OAuth 1.0的部署经验之上构建，也包括从更广泛的IETF社区收集到其他用户案例和可扩展性需求。OAuth 2.0协议不向后兼容OAuth 1.0。这两个版本可以在网络上共存，实现者可以选择同时支持他们。然而，本规范的用意是新的实现按本文档的规定支持Auth 2.0，OAuth 1.0仅用于支持现有的部署。OAuth 2.0协议与OAuth 1.0协议实现细节没有太多关联。熟悉OAuth 1.0的实现者应该理解本文档，而非对有关OAuth 2.0的结构和细节做任何假设。

Links

- 目录
- 下一节 1.1 角色

1.1. 角色

OAuth定义了四种角色：

- 资源所有者

  能够许可对受保护资源的访问权限的实体。当资源所有者是个人时，它被称为最终用户。

- 资源服务器

  托管受保护资源的服务器，能够接收和响应使用访问令牌对受保护资源的请求。

- 客户端

  使用资源所有者的授权代表资源所有者发起对受保护资源的请求的应用程序。术语“客户端”并非特指任何特定的的实现特点（例如：应用程序是否是在服务器、台式机或其他设备上执行）。

- 授权服务器

  在成功验证资源所有者且获得授权后颁发访问令牌给客户端的服务器。

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。授权服务器可以和资源服务器是同一台服务器，也可以是分离的个体。一个授权服务器可以颁发被多个资源服务器接受的访问令牌。

Links

- 目录
- 上一节 1. 简介
- 下一节 1.2 协议流程

1.2. 协议流程

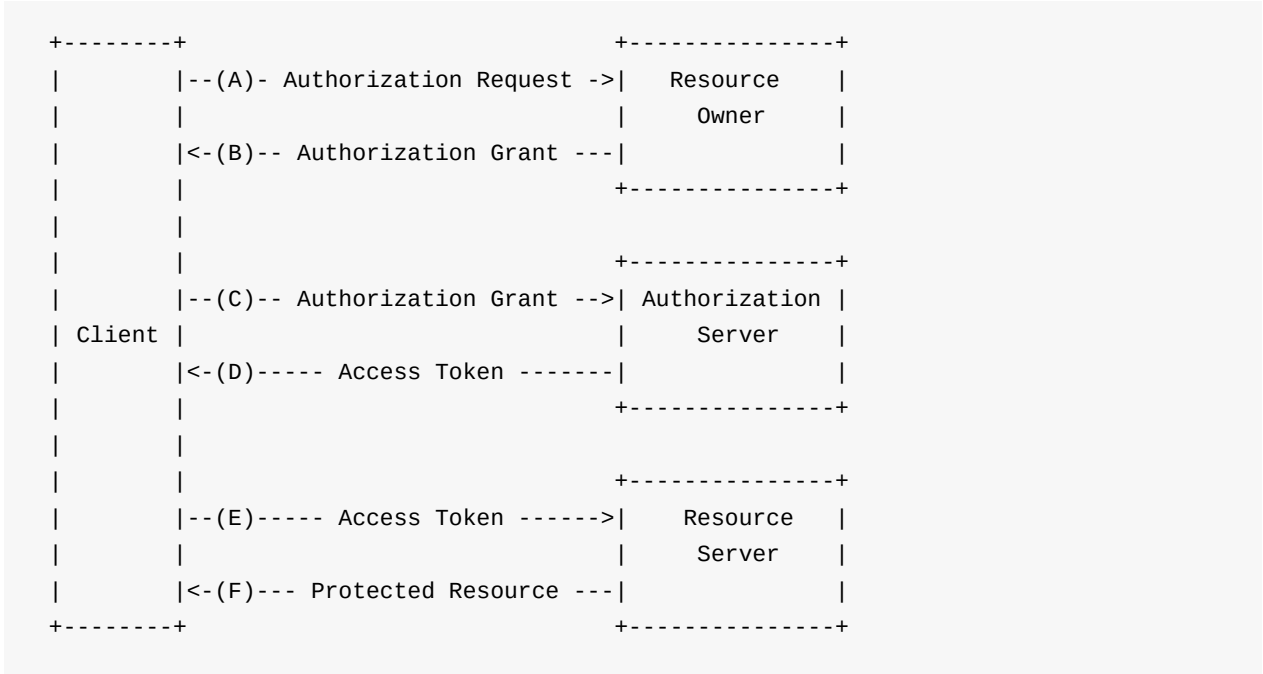


图1：抽象的协议流程

图1中所示的抽象的OAuth 2.0流程描述了四种角色之间的交互，包括以下步骤：

- （A）客户端从资源所有者处请求授权。授权请求可以直接向资源所有者发起（如图所示），或者更可取的是通过授权服务器作为中介间接发起。
- （B）客户端收到授权许可，这是一个代表资源所有者的授权的凭据，使用本规范中定义的四种许可类型之一或者使用扩展许可类型表示。授权许可类型取决于客户端请求授权所使用的方法以及授权服务器支持的类型。
- （C）客户端与授权服务器进行身份认证并出示授权许可以请求访问令牌。
- （D）授权服务器验证客户端身份并验证授权许可，若有效则颁发访问令牌。
- （E）客户端从资源服务器请求受保护资源并出示访问令牌进行身份验证。
- （F）资源服务器验证访问令牌，若有效则处理该请求。

客户端从资源所有者获得授权许可（步骤（A）和（B）所示）的更好方法是使用授权服务器作为中介，如4.1节 图3所示。

Links

- 目录
- 上一节 角色
- 下一节 授权许可

1.3. 授权许可

授权许可是一个代表资源所有者授权（访问受保护资源）的凭据，客户端用它来获取访问令牌。本规范定义了四种许可类型——授权码、隐式许可、资源所有者密码凭据和客户端凭据——以及用于定义其他类型的可扩展性机制。

Links

- 目录
- 上一节 协议流程
- 下一节 授权码

1.3.1. 授权码

授权码通过使用授权服务器做为客户端与资源所有者的中介而获得。客户端不是直接从资源所有者请求授权，而是引导资源所有者至授权服务器（由在RFC2616中定义的用户代理），授权服务器之后引导资源所有者带着授权码回到客户端。

在引导资源所有者携带授权码返回客户端前，授权服务器会鉴定资源所有者身份并获得其授权。由于资源所有者只与授权服务器进行身份验证，所以资源所有者的凭据不需要与客户端分享。

授权码提供了一些重要的安全益处，例如验证客户端身份的能力，以及向客户端直接的访问令牌的传输而非通过资源所有者的用户代理来传送它而潜在暴露给他人（包括资源所有者）。

Links

- 目录
- 上一节 授权许可
- 下一节 隐式授权

1.3.2. 隐式许可

隐式许可是为用如JavaScript等脚本语言在浏览器中实现的客户端而优化的一种简化的授权码流程。在隐式许可流程中，不再给客户端颁发授权码，取而代之的是客户端直接被颁发一个访问令牌（作为资源所有者的授权）。这种许可类型是隐式的，因为没有中间凭据（如授权码）被颁发（之后用于获取访问令牌）。

当在隐式许可流程中颁发访问令牌时，发授权服务器不对客户端进行身份验证。在某些情况下，客户端身份可以通过用于向客户端传送访问令牌的重定向URI验证。访问令牌可能会暴露给资源所有者，或者对资源所有者的用户代理有访问权限的其他应用程序。

隐式许可提高了一些客户端（例如一个作为浏览器内应用实现的客户端）的响应速度和效率，因为它减少了获取访问令牌所需的往返数量。然而，这种便利应该和采用隐式许可的安全影响作权衡，如那些在10.3和10.16节中所述的，尤其是当授权码许可类型可用的时候。

Links

- 目录
- 上一节 授权码
- 下一节 资源所有者密码授权

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），可以直接作为获取访问令牌的授权许可。这种凭据只能应该当资源所有者和客户端之间具有高度信任时（例如，客户端是设备的操作系统的一部分，或者是一个高度特权应用程序），以及当其他授权许可类型（例如授权码）不可用时被使用。

尽管本授权类型需要对资源所有者凭据直接的客户端访问权限，但资源所有者凭据仅被用于一次请求并被交换为访问令牌。通过凭据和长期有效的访问令牌或刷新令牌的互换，这种许可类型可以消除客户端存储资源所有者凭据供将来使用的需要。

Links

- 目录
- 上一节 隐式授权
- 下一节 客户端凭据

1.3.4. 客户端凭据

当授权范围限于客户端控制下的受保护资源或事先与授权服务器商定的受保护资源时客户端凭据可以被用作为一种授权许可。典型的当客户端代表自己表演（客户端也是资源所有者）或者基于与授权服务器事先商定的授权请求对受保护资源的访问权限时，客户端凭据被用作为授权许可。

Links

- 目录
- 上一节 资源所有者密码授权
- 下一节 访问令牌

1.4. 访问令牌

访问令牌是用于访问受保护资源的凭据。访问令牌是一个代表向客户端颁发的授权的字符串。该字符串通常对于客户端是不透明的。令牌代表了访问权限的由资源所有者许可并由资源服务器和授权服务器实施的具体范围和期限。

令牌可以表示一个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（即令牌字符串由数据和签名组成）。额外的身份验证凭据——在本规范范围以外——可以被要求以便客户端使用令牌。

访问令牌提供了一个抽象层，用单一的资源服务器能理解的令牌代替不同的授权结构（例如，用户名和密码）。这种抽象使得颁发访问令牌比颁发用于获取令牌的授权许可更受限制，同时消除了资源服务器理解各种各样身份认证方法的需要。

基于资源服务器的安全要求访问令牌可以有不同的格式、结构及采用的方法（如，加密属性）。访问令牌的属性和用于访问受保护资源的方法超出了本规范的范围，它们在RFC6750等配套规范中定义。

Links

- 目录
- 上一节 客户端凭据
- 下一节 刷新令牌

1.5. 刷新令牌

刷新令牌是用于获取访问令牌的凭据。刷新令牌由授权服务器颁发给客户端，用于在当前访问令牌失效或过期时，获取一个新的访问令牌，或者获得相等或更窄范围的额外的访问令牌（访问令牌可能具有比资源所有者所授权的更短的生命周期和更少的权限）。颁发刷新令牌是可选的，由授权服务器决定。如果授权服务器颁发刷新令牌，在颁发访问令牌时它被包含在内（即图1中的步骤D）。

刷新令牌是一个代表由资源所有者给客户端许可的授权的字符串。该字符串通常对于客户端是不透明的。该令牌表示一个用于检索授权信息的标识符。不同于访问令牌，刷新令牌设计只与授权服务器使用，并不会发送到资源服务器。

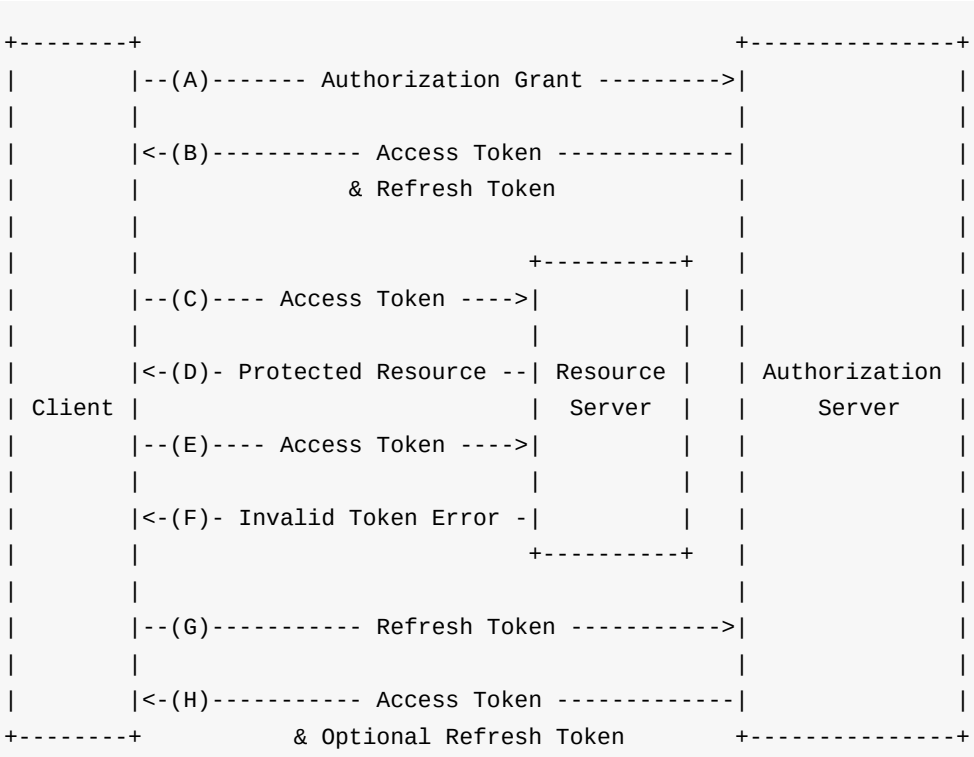


图2：刷新过期的访问令牌

图2中的所示流程包含以下步骤：

- （A）客户端通过与授权服务器进行身份验证并出示授权许可请求访问令牌。
- （B）授权服务器对客户端进行身份验证并验证授权许可，若有效则颁发访问令牌和刷新令牌。
- （C）客户端通过出示访问令牌向资源服务器发起受保护资源的请求。
- （D）资源服务器验证访问令牌，若有效则满足该要求。
- （E）步骤（C）和（D）重复进行，直到访问令牌到期。如果客户端知道访问令牌已过期，跳到步骤（G），否 则它将继续发起另一个对受保护资源的请求。

- （F）由于访问令牌是无效的，资源服务器返回无效令牌错误。
- （G）客户端通过与授权服务器进行身份验证并出示刷新令牌，请求一个新的访问令牌。客户端身份验证要求基于客户端的类型和授权服务器的策略。
- （H）授权服务器对客户端进行身份验证并验证刷新令牌，若有效则颁发一个新的访问令牌（和——可选地——一个新的刷新令牌）。

步骤（C）、（D）、（E）和（F）在本规范的范围以外，如第7节中所述。

Links

- 目录
- 上一节 访问令牌
- 下一节 TLS 版本

1.6. TLS版本

本规范任何时候使用传输层安全性（TLS），基于广泛的部署和已知的安全漏洞TLS的相应版本（或多个版本）将会随时间而变化。在本规范撰写时，TLS 1.2版RFC5246是最新的版本，但它具有非常局限的部署基础，可能还未准备好可以实现。TLS 1.0版RFC2246是部署最广泛的版本并将提供最宽泛的互操作性。

实现也可以支持满足其安全需求的其他传输层安全机制。

Links

- 目录
- 上一节 刷新令牌
- 下一节 HTTP重定向

1.7. HTTP重定向

本规范广泛采用了HTTP重定向，有此客户端或授权服务器引导资源所有者的用户代理到另一个目的地址。虽然本规范中的例子演示了HTTP 302状态码的使用，但是任何其他通过用户代理完成重定向的方法都是允许的并被考虑作为实现细节。

Links

- 目录
- 上一节 TLS版本
- 下一节 互操作性

1.8. 互操作性

OAuth 2.0提供了丰富的具有明确的安全性质的授权框架。然而，尽管在其自身看来是一个带有许多可选择组件的丰富且高度可扩展的框架，本规范有可能产生许多非可互操作的实现。

此外，本规范中留下一些必需组件部分或完全没有定义（例如，客户端注册、授权服务器性能、端点发现等）。没有这些组件，客户端必须针对特定的授权服务器和资源服务器被手动并专门配置，以进行互操作。

本框架设计具有一个明确的期望，即未来工作将定义实现完整的web范围的互操作性所需的规范性的配置和扩展。

Links

- 目录
- 上一节 HTTP重定向
- 下一节 符号约定

1.9. 符号约定

本规范中的关键词“必须”、“不能”、“必需的”、“要”、“不要”、“应该”、“不应该”、“推荐的”、“可以”以及“可选的”按RFC2119所述解释。 本规范使用RFC5234的扩展巴科斯-诺尔范式(ABNF)表示法。此外，来自“统一资源标识符（URI）：通用语法”RFC3986的规则URI引用也包含在内。

某些安全相关的术语按照RFC4949中定义的意思理解。这些术语包括但不限于：“攻击”、“身份认证”、“授权”、“证书”、“机密”，“凭据”，“加密”，“身份”，“记号”，“签名”，“信任”，“验证”和“核实”。

除非另有说明，所有协议参数的名称和值是大小写敏感的。

Links

- 目录
- 上一节 互操作性
- 下一节 客户端注册

2.0 客户端注册

在开始协议前，客户端在授权服务器注册。客户端在授权服务器上注册所通过的方式超出了本规范，但典型的涉及到最终用户与HTML注册表单的交互。

客户端注册不要求客户端与授权服务器之间的直接交互。在授权服务器支持时，注册可以依靠其他方式来建立信任关系并获取客户端的属性（如重定向URI、客户端类型）。例如，注册可以使用自发行或第三方发行声明或通过授权服务器使用信任通道执行客户端发现完成。

当注册客户端时，客户端开发者应该：

- 指定2.1节所述的客户端类型，
- 提供它的如3.1.2节所述的客户端重定向URI，且
- 包含授权服务器要求的任何其他信息（如，应用名称、网址、描述、Logo图片、接受法律条款等）。

2.1. 客户端类型

根据客户端与授权服务器安全地进行身份验证的能力（即维护客户端凭据机密性的能力），OAuth定义了两种客户端类型：

- 机密客户端
  能够维持其凭据机密性（如客户端执行在具有对客户端凭据有限访问权限的安全的服务器上），或者能够使用 其他方式保证客户端身份验证的安全性。
- 公开客户端
  不能够维持其凭据的机密性（如客户端执行在由资源所有者使用的设备上，例如已安装的本地应用程序或基于Web浏览器的应用），且不能通过其他方式保证客户端身份验证的安全性。 客户端类型的选择基于授权服务器的安全身份认证定义以及其对客户端凭据可接受的暴露程度。授权服务器不应该对客户端类型做假设。

客户端可以以分布式的组件集合实现，每一个组件具有不同的客户端类型和安全上下文（例如，一个同时具有机密的基于服务器的组件和公开的基于浏览器的组件的分布式客户端）。如果授权服务器不提供对这类客户端的支持，或不提供其注册方面的指导，客户端应该注册每个组件为一个单独的客户端。 本规范围绕下列客户端配置涉及：

- Web应用程序
  Web应用是一个运行在Web服务器上的机密客户端。资源所有者通过其使用的设备上的用户代理里渲染的HTML用户界面访问客户端。客户端凭据以及向客户端颁发的任何访问令牌都存储在Web服务器上且不会暴露给资源所有者或者被资源所有者可访问。
- 基于用户代理的应用
  基于用户代理的应用是一个公开客户端，客户端的代码从Web服务器下载，并在资源所有者使用的设备上的用户代理（如Web浏览器）中执行。协议数据和凭据对于资源所有者是可轻易访问的（且经常是可见的）。由于这些应用驻留在用户代理内，在请求授权时它们可以无缝地使用用户代理的功能。
- 本机应用程序
  本机应用是一个安装、运行在资源所有者使用的设备上的公开客户端。协议数据和凭据对于资源所有者是可访问的。假定包含在应用程序中的任何客户端身份认证凭据可以被提取。另一方面，动态颁发的如访问令牌或者刷新令牌等凭据可以达到可接受的保护水平。至少，这些凭据被保护不被应用可能与之交互的恶意服务器接触。在一些平台上，这些凭证可能被保护免于被驻留在相同设备上的其他应用接触。

2.2. 客户端标识

授权服务器颁发给已注册客户端客户端标识——一个代表客户端提供的注册信息的唯一字符串。客户端标识不是一个秘密，它暴露给资源所有者并且不能单独用于客户端身份验证。客户端标识对于授权服务器是唯一的。

客户端的字符串大小本规范未定义。客户端应该避免对标识大小做假设。授权服务器应记录其发放的任何标识的大小。

2.3. 客户端身份验证

如果客户端类型是机密的，客户端和授权服务器建立适合于授权服务器的安全性要求的客户端身份验证方法。授权服务器可以接受符合其安全要求的任何形式的客户端身份验证。

机密客户端通常颁发（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/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。然而，授权服务器不能依靠公共客户端身份验证达到识别客户端的目的。

客户端在每次请求中不能使用一个以上的身份验证方法。

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基本身份验证方案与授权服务器进行身份验证。客户端标识使用按照附录B的“application/x-www-form-urlencoded”编码算法编码，编码后的值用作用户名；客户端密码使用相同的算法编码并用作密码。授权服务器必须支持HTTP基本身份验证方案，用于验证被颁发客户端密码的客户端的身份。例如（额外的换行仅用于显示目的）：

```
Authorization: Basic czZCaGRSa3F0Mzo3RmpmcDBaQnIxS3REUmJuZlZkbUl3
```

此外，授权服务器可以使用下列参数支持在请求正文中包含客户端凭据：

- client_id
  必需的。如2.2节所述的注册过程中颁发给客户端的客户端标识。
- client_secret
  必需的。客户端秘钥。 客户端可以忽略该参数若客户端秘钥是一个空字符串。

使用这两个参数在请求正文中包含客户端凭据是不被建议的，应该限于不能直接采用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。参数只能在请求正文中传送，不能包含在请求URI中。

例如，使用请求正文参数请求刷新访问令牌（第6节）（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=refresh_token&refresh_token=tGzv3JOkF0XG5Qx2TlKWIA&client_id=s6BhdRkqt3&cl
ient_secret=7Fjfp0ZBr1KtDRbnfVdmIw
```

当使用密码身份验证发送请求时，授权服务器必须要求使用如1.6所述的TLS。

由于该客户端身份验证方法包含密码，授权服务器必须保护所有使用到密码的端点免受暴力攻击。

2.3.2. 其他身份验证方法

授权服务器可以支持任何与其安全要求匹配的合适的HTTP身份验证方案。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（注册记录）和认证方案之间的映射。

2.4. 未注册客户端

本规范不排除使用未注册的客户端。然而，使用这样的客户端超出了本规范的范围，并需要额外的安全性分析并审查其互操作性影响。

3. 协议端点

授权过程采用了两种授权服务器端点（HTTP资源）：

- 授权端点——客户端用其通过用户代理重定向从资源所有者获取授权。
- 令牌端点——客户端用其将授权许可交换为访问令牌，通常伴有客户端身份验证。

以及一种客户端端点：

- 重定向端点——授权服务器用其通过资源所有者用户代理向客户端返回含有授权凭据的响应。

并不是每种授权许可类型都采用两种端点。

扩展许可类型可以按需定义其他端点。

3.1. 授权端点

授权端点用于与资源所有者交互获取授权许可。 授权服务器必须先验证资源所有者的身份。 授权服务器对资源所有者进行身份验证的方式（例如，用户名和密码登录、会话cookies）超出了本规范的范围。

客户端通过何种方式获得授权端点的位置超出了本文档范围，但该位置通常在服务文档中提供。

端点URI可以包含“application/x-www-form-urlencoded”格式（按附录B）的查询部分（RFC3986的3.4节），当添加额外的查询参数时必须保留该部分。端点URI不得包含片段部分。

由于向授权端点的请求引起用户身份验证和明文凭据传输（在HTTP响应中），当向授权端点发送请求时，授权服务器必须要求如1.6节所述的TLS的使用。

授权服务器对于授权端点必须支持使用HTTP“GET”方法RFC2616，也可以支持使用“POST”的方法。

发送的没有值的参数必须被对待为好像它们在请求中省略了。授权服务器必须忽略不能识别的请求参数。 请求和响应参数不能包含超过一次。

3.1.1. 响应类型

授权端点被授权码许可类型和隐式许可类型流程使用。客户端使用下列参数通知授权服务器期望的许可类型：

- response_type
  必需的。其值必须是如4.1.1节所述用于请求授权码的“code”，如4.2.1节所述用于请求访问令牌的“token”（隐式许可）或者如8.4节所述的一个注册的扩展值之中的一个。

扩展响应类型可以包含一个空格（%x20）分隔的值的列表，值的顺序并不重要（例如，响应类型“a b”与“b a”相同）。 这样的复合响应类型的含义由他们各自的规范定义。

如果授权请求缺少“response_type”参数，或者如果响应类型不被理解，授权服务器必须返回一个4.1.2.1所述的错误响应。

3.1.2. 重定向端点

在完成与资源所有者的交互后，授权服务器引导资源所有者的用户代理返回到客户端。授权服务器重定向用户代理至客户端的重定向端点，该端点是事先在客户端注册过程中或者当发起授权请求时与授权服务器建立的。

重定向端点URI必须是如RFC3986的3.4节所述的绝对URI。端点URI可以包含“application/x-www-form-urlencoded”格式（按附录B）的查询部分（RFC3986的3.4节），当添加额外的查询参数时必须保留该部分。端点URI不得包含片段部分。

3.2. 令牌端点

客户端通过提交它的授权许可或刷新令牌使用令牌端点获取访问令牌。令牌端点被用于除了隐式许可类型（因为访问令牌是直接颁发的）外的每种授权许可中。

客户端通过何种方式获得令牌端点的位置超出了本文档范围，但该位置通常在服务文档中提供。

端点URI可以包含“application/x-www-form-urlencoded”格式（按附录B）的查询部分（RFC3986的3.4节），当添加额外的查询参数时必须保留该部分。端点URI不得包含片段部分。

由于向令牌端点的请求引起明文凭据的传输（在HTTP请求和响应中），当向令牌端点发送请求时，授权服务器必须要求如1.6节所述的TLS的使用。

当发起访问令牌请求时，客户端必须使用HTTP“POST”方法。

发送的没有值的参数必须被对待为好像它们在请求中省略了。授权服务器必须忽略不能识别的请求参数。请求和响应参数不能包含超过一次。

- 3.2.1. 客户端身份验证

3.2.1. 客户端身份验证

在向令牌端点发起请求时，机密客户端或其他被颁发客户端凭据的客户端必须如2.3节所述与授权服务器进行身份验证。客户端身份验证用于：

- 实施刷新令牌和授权码到它们被颁发给的客户端的绑定。当授权码在不安全通道上向重定向端点传输时，或者 当重定向URI没有被完全注册时，客户端身份验证是关键的。
- 通过禁用客户端或者改变其凭据从被入侵的客户端恢复，从而防止攻击者滥用被盗的刷新令牌。改变单套客户端凭据显然快于撤销一整套刷新令牌。
- 实现身份验证管理的最佳实践，要求定期凭证轮转。轮转一整套刷新令牌可能是艰巨的，而轮转单组客户端凭据显然更容易。

在向令牌端点发送请求时，客户端可以使用“client_id”请求参数标识自己。向令牌端点的“authorization_code”和“grant_type”请求中，未经身份验证的客户端必须发送它的“client_id”，以防止自己无意中接受了本打算给具有另一个“client_id”的客户端的代码。这保护了客户端免于被替换认证码。（它没有对手保护起源提供额外的安全。）

3.3. 访问令牌范围

授权端点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”响应参数通知客户端颁发的访问令牌的范围。

范围参数的值表示为以空格分隔，大小写敏感的字符串。 由授权服务器定义该字符串。如果该值包含多个空格分隔的字符串，他们的顺序并不重要且每个字符串为请求的范围添加一个额外的访问区域。

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

基于授权服务器的策略或资源拥有者的指示，授权服务器可以全部或部分地忽略客户端请求的范围。如果颁发的访问令牌范围和客户端请求的范围不同，授权服务器必须包含“scope”响应参数通知客户端实际许可的范围。

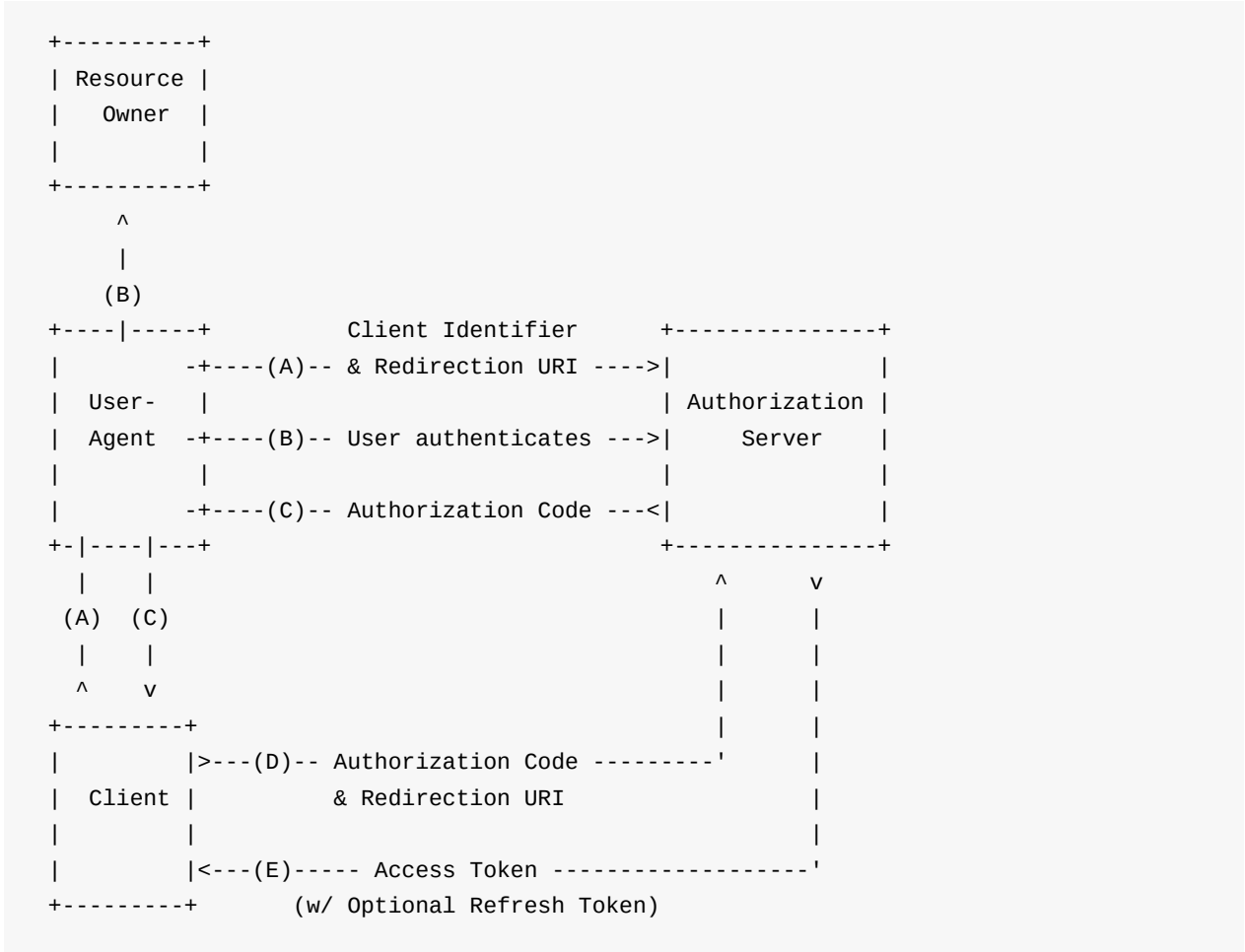
在请求授权时如果客户端忽略了范围参数，授权服务器必须要么使用预定义的默认值处理请求，要么使请求失败以指出无效范围。授权服务器应该记录它的范围需求和默认值（如果已定义）。

4. 获得授权

为了请求访问令牌，客户端从资源所有者获得授权。授权表现为授权许可的形式，客户端用它请求访问令牌。OAuth定义了四种许可类型：授权码、隐式许可、资源所有者密码凭据和客户端凭据。它还提供了扩展机制定义其他许可类型。

4.1. 授权码许可

授权码许可类型用于获得访问令牌和刷新令牌并且为受信任的客户端进行了优化。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）。



注：说明步骤（A）、（B）和（C）的直线因为通过用户代理而被分为两部分。
图3：授权码流程

在图3中所示的流程包括以下步骤：

- （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包括授权码和之前客户端提供的任何本地状态。

- （D）客户端通过包含上一步中收到的授权码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。客户端包含用于获得授权码的重定向URI来用于验证。
- （E）授权服务器对客户端进行身份验证，验证授权代码，并确保接收的重定向URI与在步骤（C）中用于重定向（资源所有者的用户代理）到客户端的URI相匹配。如果通过，授权服务器响应返回访问令牌与可选的刷新令牌。

4.1.1. 授权请求

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。值必须被设置为“code”。
- client_id
  必需的。如2.2节所述的客户端标识。
- redirect_uri
  可选的。如3.1.2节所述。
- scope
  可选的。如3.3节所述的访问请求的范围。
- state
  推荐的。客户度用于维护请求和回调之间的状态的不透明的值。当重定向用户代理回到客户端时，授权服务器包含此值。该参数应该用于防止如10.12所述的跨站点请求伪造。

客户端使用HTTP重定向响应向构造的URI定向资源所有者，或者通过经由用户代理至该URI的其他可用方法。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=code&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。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。

4.1.2. 授权响应

如果资源所有者许可访问请求，授权服务器颁发授权码，通过按附录B使用“application/x-www-form-urlencoded”格式向重定向URI的查询部分添加下列参数传递授权码至客户端：

- code
  必需的。授权服务器生成的授权码。授权码必须在颁发后很快过期以减小泄露风险。推荐的最长的授权码生命周期是10分钟。客户端不能使用授权码超过一次。如果一个授权码被使用一次以上，授权服务器必须拒绝该请求并应该撤销（如可能）先前发出的基于该授权码的所有令牌。授权码与客户端标识和重定向URI绑定。
- state
  必需的，若“state”参数在客户端授权请求中提交。从客户端接收的精确值。

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?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。

- 4.1.2.1. 错误响应

4.1.3. 访问令牌请求

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，如果客户端没有如3.2.1节所述与授权服务器进行身份认证。

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 必需的，如果客户端没有如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%2Fc
lient%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，
- 若包括了客户端身份验证，验证客户端身份，
- 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所述包含在初始授权请求中，且若包含，确保它们的值是相同的。

4.1.4. 访问令牌响应

如果访问令牌请求是有效的且被授权，授权服务器如5.1节所述颁发访问令牌以及可选的刷新令牌。如果请求客户端身份验证失败或无效，授权服务器如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,
  "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
  "example_parameter":"example_value"
}
```

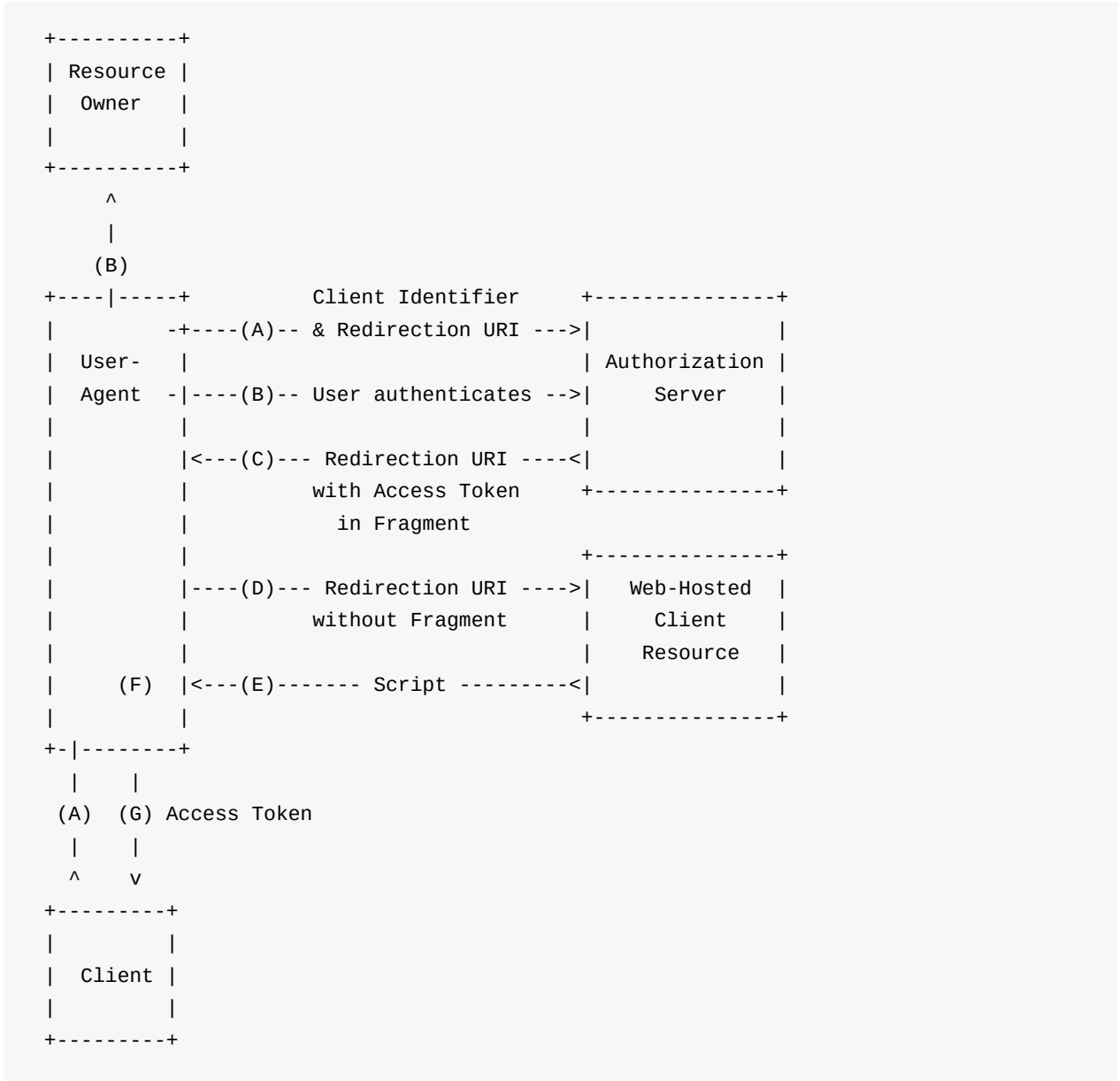
4.2. 隐式许可

隐式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许可类型，客户端收到访问令牌作为授权请求的结果。

隐式许可类型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。



注：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：隐式许可流程

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1.3.2节和第9节了解使用隐式许可的背景。

参见10.3节和10.16节了解当使用隐式许可时的重要安全注意事项。

4.2.1. 授权请求

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。如2.2节所述的客户端标识。
- redirect_uri
  可选的。如3.1.2节所述。
- scope
  可选的。如3.3节所述的访问请求的范围。
- state
  推荐的。客户度用于维护请求和回调之间的状态的不透明的值。当重定向用户代理回到客户端时，授权服务器包含此值。该参数应该用于防止如10.12所述的跨站点请求伪造。

客户端使用HTTP重定向响应向构造的URI定向资源所有者，或者通过经由用户代理至该URI的其他可用方法。

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%3
A%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。

4.2.2. 访问令牌响应

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3.3节所述的访问令牌的范围。
- state
  必需的，若“state”参数在客户端授权请求中提交。从客户端接收的精确值。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_ty
pe=example&expires_in=3600
```

开发人员应注意，一些用户代理不支持在HTTP“Location”HTTP响应标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。

- 4.2.2.1. 错误响应

4.3. 资源所有者密码凭据许可

资源所有者密码凭据许可类型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。

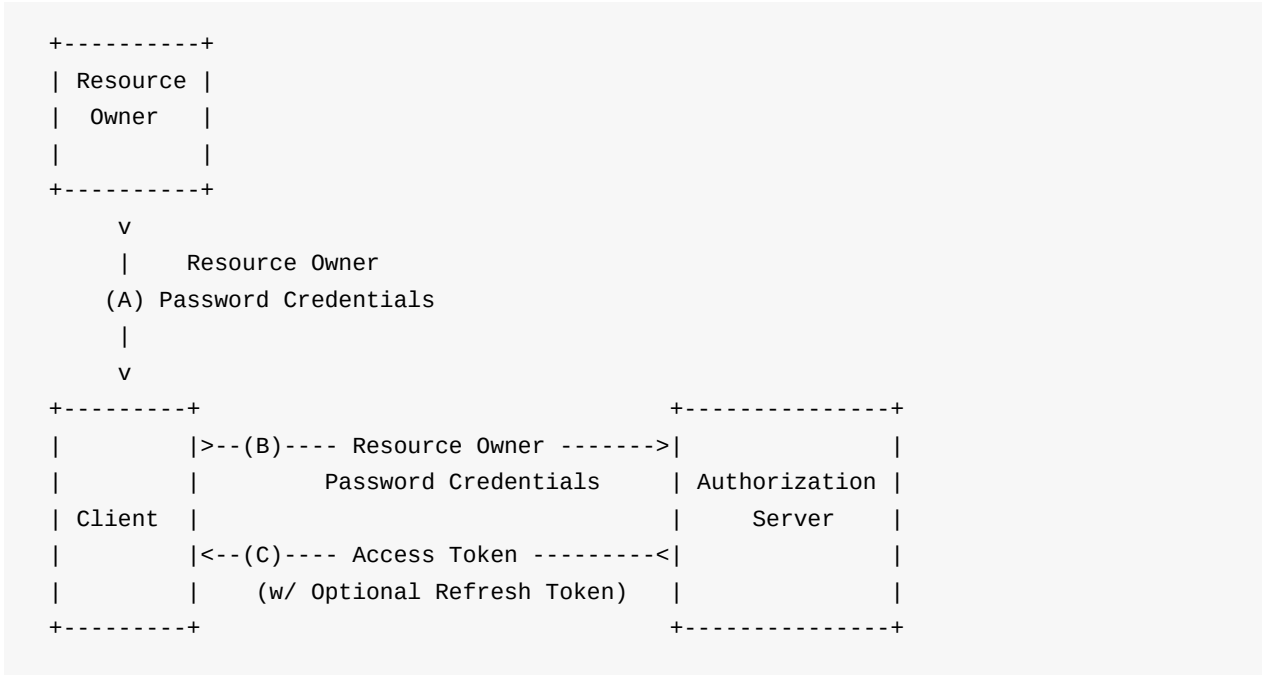


图5：资源所有者密码凭据流程

图5中的所示流程包含以下步骤：

- （A）资源所有者提供给客户端它的用户名和密码。
- （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。

4.3.1. 授权请求和响应

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。

4.3.2. 访问令牌请求

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）。

4.3.3. 访问令牌响应

如果访问令牌请求是有效的且被授权，授权服务器如5.1节所述颁发访问令牌以及可选的刷新令牌。如果请求客户端身份验证失败或无效，授权服务器如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,
  "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
  "example_parameter":"example_value"
}
```

4.4. 客户端凭据许可

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：客户端凭证流程

图6中的所示流程包含以下步骤：

- （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。

4.4.1. 授权请求和响应

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。

4.4.2. 访问令牌请求

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3.3节所述的访问请求的范围。

客户端必须如3.2.1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。

4.4.3. 访问令牌响应

如果访问令牌请求是有效的且被授权，授权服务器如5.1节所述颁发访问令牌以及可选的刷新令牌。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"
}
```

4.5. 扩展许可

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%3Asaml2bearer&assertion=PEFzc2VydG
lvbiBJc3N1ZUluc3RhbnQ9IjIwMTEtMDU[...为简洁起见省略...]aG5TdGF0ZW1lbnQ-PC9Bc3NlcnRpb24-
```

如果访问令牌请求是有效的且被授权，授权服务器如5.1节所述颁发访问令牌以及可选的刷新令牌。如果请求因客户端身份验证失败或无效，授权服务器如5.2节所述的返回错误响应。

5. 颁发访问令牌

如果访问令牌请求是有效的且被授权，授权服务器如5.1节所述颁发访问令牌以及可选的刷新令牌。如果请求因客户端身份验证失败或无效，授权服务器如5.2节所述的返回错误响应。

5.1. 成功的响应

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。

5.2. 错误响应

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。下列ASCII[USASCII]错误代码之一：
  - 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”请求标头域进行身份验证，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”响应标头字段。
  - 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。提供额外信息的人类可读的ASCII[USASCII]文本，用于协助客户端开发人员理解所发生的错误。“error_description”参数的值不能包含集合%x20-21 /%x23-5B /%x5D-7E以外的字符。
- error_uri
  可选的。指向带有有关错误的信息的人类可读网页的URI，用于提供客户端开发人员关于该错误的额外信息。“error_uri”参数值必须符合URI参考语法，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"
}
```

6. 刷新访问令牌

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=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。

7. 访问受保护资源

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”的请求标头字段RFC2617。

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，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 F_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="kDZvddkndxvhGRXZhvuD
jEWhGeE="
```

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。

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。

8. 可扩展性

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
```

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。

8.3. 定义新的授权许可类型

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。

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。

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许可错误响应（4.1.2.1节）、隐式许可错误响应（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
```

9. 本机应用程序

本机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，本机移动应用）。本机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。本机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-本机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，反过来使响应对本机应用程序可用。
- 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，本机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）。 当在隐式许可类型和授权码许可类型中选择时，下列应该被考虑：
- 使用授权码许可类型的本机应用程序应该这么做而不需使用用户凭据，因为本机应用程序无力保持客户端凭据的机密性。
- 当使用隐式许可类型流程时，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。

10. 安全考量

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。

10.1. 客户端身份验证

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。

10.2. 客户端仿冒

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。

10.3. 访问令牌

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。

当使用隐式授权许可类型时，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。

10.4. 刷新令牌

授权服务器可以给Web应用客户端和本机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。

10.5. 授权码

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参照标头被泄露。

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。

10.6. 授权码重定向URI伪造

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。

10.7. 资源所有者密码凭据

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。

10.8. 请求机密性

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。

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。

10.10. 凭据猜测攻击

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。

10.11. 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。

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。

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“授权”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机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”标头实施。 该标头可以有两个值，“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。

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。

10.15. 自由重定向器

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。

10.16. 在隐式流程中滥用访问令牌假冒资源所有者

对于使用隐式流程的公共客户端，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。

在隐式流程（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机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隐式流程（例如，限制访问令牌的受众）。

11. IANA考量

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。

11.1.1. 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。

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。

11.2.2. 最初的注册表内容

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

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。

11.3.1. 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

11.3.2. 最初的注册表内容

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

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。

11.4.1. 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。可能的位置是授权代码许可错误响应（4.1.2.1节），隐式许可错误响应（4.2.2.1节），令牌错误响应（5.2节），或资源访问错误的响应（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。

12. 参考文献

12.1. 规范性参考文件

- [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, http://www.w3.org/TR/2008/REC-xml-20081126.

12.2. 参考性引用文献

- [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.

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引用”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-20081126]2.2节中定义的字符，但忽略了回车和换行字符。）

A.1. “client_id”语法

“client_id”元素在2.3.1节定义：

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

A.2. “client_secret”语法

“client_secret”元素在2.3.1节定义：

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

A.3. “response_type”语法

“response_type”元素在3.1.1节和8.4节中定义：

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

A.4. “scope”语法

“scope”元素在3.3节中定义：

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

A.5. “state”语法

“state”元素在4.1.1、4.1.2、4.1.2.1、4.2.1、4.2.2和4.2.2.1节中定义：

```
state      = 1*VSCHAR
```

A.6. “redirect_uri”语法

“redirect_uri”元素在4.1.1、4.1.3、4.2.1定义：

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

A.7. “error”语法

“error”元素在4.1.2.1、4.2.2.1、5.2、7.2和8.5中定义：

```
error             = 1*NQSCHAR
```

A.8. “error_description”语法

“error_description”元素在4.1.2.1、4.2.2.1、5.2和7.2中定义：

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

A.9. “error_uri”语法

“error_uri”元素在4.1.2.1、4.2.2.1、5.2和7.2中定义：

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

A.10. “grant_type”语法

“grant_type”元素在4.1.3、4.3.2、4.4.2、4.5和6中定义：

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

A.11. “code”语法

“code”元素在4.1.3节中定义：

```
code        = 1*VSCHAR
```

A.12. “access_token”语法

“access_token”元素在4.2.2节和5.1节中定义：

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

A.13. “token_type”语法

“token_type”元素在4.2.2、5.1和8.1中定义：

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

A.14. “expires_in”语法

“expires_in”元素在4.2.2节和5.1节中定义的：

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

A.15. “username”语法

“username”元素在4.3.2节中定义：

```
username = *UNICODECHARNOCRLF
```

A.16. “password”语法

“password”元素在4.3.2节中定义：

```
password = *UNICODECHARNOCRLF
```

A.17. “refresh_token”语法

“refresh_token”元素在5.1节和第6节中定义：

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

A.18. 端点参数语法

新的端点参数的语法在8.2节中定义：

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

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), (6) U+20AC (EURO SIGN)
```

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
```

附录C. 致谢

最初的OAuth 2.0协议规范是由David Recordon编辑的，基于先前的两个发行版：OAuth 1.0社区规范RFC5849和OAuth WRAP（OAuth Web资源授权配置）[OAuth WRAP]。Eram Hammer然后编辑了包含在此RFC中的中间草稿。Torsten Lodderstedt, Mark McGloin, Phil Hunt, Anthony Nadalin和John Bradley起草了“安全注意事项”一节。使用“application/x-www-form.urlencoded”媒体类型一节由Julian Reschke起草。ABNF一节由Julian Reschke起草。

OAuth 1.0社区规范由Eran Hammer编辑并由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以及Andy Smith撰写。

OAuth WRAP规范由 Dick Hardt编辑，由Brian Eaton, Yaron Y. Goland, Dick Hardt, and Allen Tom撰写。 本规范是OAuth工作组的工作，其中包括几十个积极的和专门的参与者。特别是，下列个人贡献了想法，反馈和措辞，格式化了最终的规范：

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

本文档由主席团Blaine Cook, Peter Saint-Andre, Hannes Tschofenig, Barry Leiba和Derek Atkins出品。

区域指导包括 Lisa Dusseault, Peter Saint-Andre和Stephen Farrell.

作者署名

Dick Hardt (editor)
Microsoft

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

- 3446
- 3500

勘误编号：3446

状态：已核实

类型：编辑

报告者：Nov Matake

报告日期：2013-01-07

审核者：Stephen Farrell

审核日期：2013-03-16

第1节：

— 资源所有者不能撤销某个第三方的访问权限而不影响其它第三方，并且必须更改第三方的密码才能做到。

应该是：

— 资源所有者不能撤销某个第三方的访问权限而不影响其它第三方，并且必须更改他们的密码才能做到。

备注：

文本原来是“他们的”但是在最后的草稿和RFC之间改成了“第三方的”。然而，“他们的”意思是“资源所有者的”，而不是“第三方的”。

勘误编号：3500

状态：已核实

类型：编辑

报告者：John Field

报告日期：2013-02-26

审核者：Stephen Farrell

审核日期：2013-03-16

第4.1节：

- （E）授权服务器对客户端进行身份验证，验证授权代码，并确保接收的重定向URI与在步骤（C）中用于重定向客户端的URI相匹配。如果通过，授权服务器响应返回访问令牌与可选的刷新令牌。

应该是：

- （E）授权服务器对客户端进行身份验证，验证授权代码，并确保接收的重定向URI与在步骤（C）中用于重定向（资源所有者的用户代理）到客户端的URI相匹配。如果通过，授权服务器响应返回访问令牌与可选的刷新令牌。

备注：

问题中的URI是用于重定向资源所有者的用户代理回到客户端传送代码的URI。（E）步骤中原文好像说该URI用于重定向客户端，但是我认为这是对“客户端”这个词的不明确／不准确的使用。OAuth客户端不会使用URI被重定向，资源所有者的用户代理会被重定向到客户端。

插入的“（资源所有者的用户代理）”会更准确但是可能会太冗长了。我觉得，至少，我们必须说“……该URI用于在步骤（C）中重定向到客户端。”