(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 103460215 A
(43)申请公布日 2013.12.18

(21)申请号 201180069099.5
(22)申请日 2011.11.04
(30)优先权数据
61/450,376 2011.03.08 US
61/452,262 2011.03.14 US
(85)PCT申请进入国家阶段日
2013.09.09
(86)PCT申请的申请数据
PCT/EP2011/005569 2011.11.04
(87)PCT申请的公布数据
WO2012/119620 EN 2012.09.13
(71)申请人 电话有限公司
地址 西班牙马德里
(72)发明人 乔治·洛伦佐 大卫·洛扎诺
迭戈·冈萨雷斯 大卫·韦森特
(74)专利代理机构 北京万慧达知识产权代理有限公司 11111
代理人 白华胜 段晓玲
(51)Int.Cl.
G06F 21/00(2013.01)

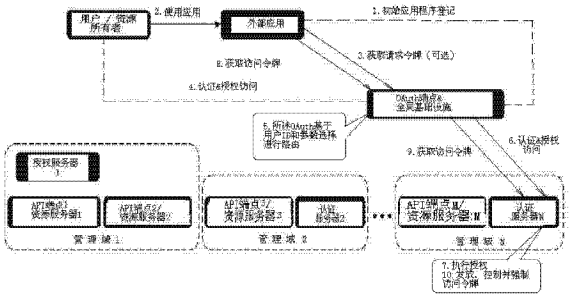
权利要求书3页 说明书11页 附图11页

(54)发明名称

为服务应用提供授权访问以便使用最终用户的受保护资源的方法

(57)摘要

一种为服务应用提供授权访问以便使用最终用户的受保护资源的方法。所述受保护资源，通常是由多个管理域的端点公布的具有代表性的API。所述服务应用事先不知道所述端点，并且所述方法还包括：i. 使用中间媒介或全局实体，用于：a) 基于灵活规则从所述管理域中选择一个管理域(即，至少基于所述最终用户的身份，但也考虑到各种用户和服务的参数来选择)；以及b) 通过开放协议，执行所述选定的管理域的安全授权允许访问所述最终用户。以及ii. 为所述服务应用提供所述选定的管理域，一旦执行了所述安全授权，通过由所述中间媒介实体建立的端点直接或经由代理访问所述用户的受保护资源。



CN 103460215 A

1. 一种为服务应用提供授权访问以便使用最终用户的受保护资源的方法，所述受保护资源由多个管理域的端点公布，所述多个管理域中的每个管理域负责分别发放和控制由属于所述管理域的最终用户允可的多个受保护资源的授权访问的后续使用，其特征在于，所述服务应用事先不知道所述端点，并且所述方法还包括：

i. 使用中间媒介或全局实体来：

a) 基于至少包括所述最终用户身份的灵活规则，从所述多个管理域中选择一个管理域；以及

b) 所述选定的管理域执行安全授权，以便通过开放协议允许访问所述最终用户；

以及

ii. 一旦执行了所述安全授权，所述选定的管理域为所述服务应用提供：通过由所述中间媒介或全局实体建立的端点之一直接或代理访问所述受保护资源。

2. 根据权利要求 1 所述的方法，其中所述受保护的资源由应用编程接口或 API 公布，所述开放协议是 OAuth 和 / 或属于服务提供商的每个所述管理域。

3. 根据权利要求 2 所述的方法，其中所述中间媒介或全局实体包括：

-OAuth 协议端点，该 OAuth 协议端点事先已被所述服务应用已知或发现，并且被所述服务应用用来触发所述 OAuth 程序；以及

- 全局的基础设施，该设施支持申请注册和证书管理。

4. 根据权利要求 3 所述的方法，包括通过所述 OAuth 过程执行所述安全授权的步骤 1.i.b)，所述 OAuth 程序至少包括：

- 所述最终用户的认证阶段；以及

- 为所述服务应用提供访问令牌以便允许访问所述受保护的资源。

5. 根据权利要求 3 或 4 所述的方法，包括所述中间媒介或全局实体，根据所述最终用户的信息和所述 OAuth 认证阶段期间提供的首选项，通过执行所述选择步骤 1.i.a)，将所述的 OAuth 程序路由到所述管理域。

6. 根据权利要求 5 所述的方法，包括通过链接页面管理所述中间媒介或全局实体与所述最终用户之间的交互。

7. 根据权利要求 4、5 或 6 所述的方法，包括所述中间或全局实体，基于所述访问令牌，路由从所述应用发送到所述管理域的 API 授权请求。

8. 根据权利要求 7 所述的方法，包括在上述认证阶段期间，使用访问令牌作为路由决策的指针，并且，所述中间媒介或全局实体提供数据库，该数据库将访问令牌与所述多个管理域中的一个管理域相关联。

9. 根据权利要求 8 所述的方法，包括通过重定向执行所述 API 授权请求的路由，所述重定向基于所述访问令牌查询所述数据库，为所述应用提供管理域的统一资源定位符或 URL，其中所述应用通过管理域的 URL 和通向受保护资源的路径访问所述受保护的资源 . 。

10. 根据权利要求 8 所述的方法，包括通过代理执行所述 API 的授权请求的路由，所述代理提取所述访问令牌并从所述全局基础设施获得与所述访问令牌相关联的所述 API 端点。

11. 根据权利要求 4 至 10 中任一项所述的方法，其中所述多个管理域使用一个公共的 web 或 wap 页面，所述页面允许与最终用户的交互，以便执行所述认证阶段的至少一部分。

12. 根据权利要求 4 至 10 中任一项所述的方法，其中所述多个管理域中的每一个都使用本地 web 或 wap 页面，该页面允许与最终用户交互，以便执 行所述认证阶段的至少一部分。

13. 根据权利要求 2 至 12 中任一项所述的方法，其中所述开放式协议是 OAuth1.0，所述 OAuth1.0 包括为所述应用授予请求令牌，所述请求令牌被用来在认证阶段期间执行询问所述最终用户来授权所述应用访问所述 API。

14. 根据权利要求 13 所述的方法，包括所述服务应用从中间媒介或全局实体得到所述请求令牌。

15. 根据权利要求 14 所述的方法，包括所述中间媒介或全局实体从所述管理域获得所述请求令牌，或者将所述请求令牌设置到管理域中。

16. 根据引用权利要求 6 的权利要求 14 或 15 所述的方法，，其中所述 OAuth 程序还包括：

- 所述应用获取所述请求令牌；

- 将所述用户转发到所述链接页面，以便找出所述管理域；

- 所述中间媒介或全局实体使用所述灵活规则来找出所述的管理域，并将所述最终用户转发到对应的公共或本地 Web 或 WAP 页面；

- 当没有其他方式的预先认证时，所述用户为所述公共或本地 Web 或 WAP 页面提供证书以便认证阶段的执行；

- 所述公共或本地 Web 或 WAP 页面将所述认证请求与所述证书转发到管理域的授权服务器；

- 所述最终用户授权所述应用访问受保护的资源；

- 所述管理域的所述授权服务器生成授权码，并通过所述中间媒介或全局实体将授权码发送到所述应用；

- 所述中间媒介或全局实体从上述管理域获得访问令牌，所述应用通过所述中间媒介或全局实体请求并获得访问令牌；

- 所述应用使用访问令牌访问所述保护资源。

17. 根据权利要求 4 至 12 中任一项所述的方法，其中所述开放协议是 OAuth2.0。

18. 根据权利要求 17 所述的方法，其中所述的 OAuth2.0 程序还包括：

- 将所述最终用户转发到所述链接页面，以便找出所述管理域；

- 所述中间媒介或全局实体使用所述灵活规则来找出适合的管理域，并将所述最终用户转发到相应的适合的公共或本地 Web 或 WAP 页面；

- 在没有以其他方式预先认证时，所述用户为所述公共或本地 Web 或 WAP 页面提供证书，以便认证阶段的执行；

- 所述公共或本地 Web 或 WAP 页面将所述认证请求与所述证书转发到管理域的授权服务器；

- 所述最终用户授权所述应用访问受保护的资源；

- 所述授权服务器生成授权码或访问令牌，并通过所述中间媒介或全局实体将所述授权码或所述访问令牌发送到所述服务应用；

- 在接收所述授权码的情况下，所述应用通过所述中间媒介或全局实体请求并获得访

问令牌，其中所述中间媒介或全局实体从上述管理域获得访问令牌；以及

- 所述应用使用所述访问令牌访问所述受保护资源。

19. 根据权利要求 4 至 18 中任一项所述的方法，包括所述授权服务器按所述最终用户的要求撤销访问令牌，或根据所述 API 的特定策略撤销所述访问令牌。

20. 根据权利要求 19 所述的方法，包括所述最终用户向所述公共或本地的页面请求撤销所述访问令牌，并且，通过所述中间媒介或全局实体将所述请求转发到所述管理域的所述授权服务器。

21. 根据权利要求 19 或 20 所述的方法，其中所述中间实体定期地检索从所述 API 端点撤销的访问令牌列表。

为服务应用提供授权访问以便使用最终用户的受保护资源的方法

技术领域

[0001] 本发明通常涉及：为服务应用提供授权访问以便使用最终用户的受保护资源的方法，典型地，所述受保护的资源是API，并且被多个的管理域的端点公开，并且所述授权访问通过OAuth程序来执行，并且更具体为一种方法，该方法包括使用中间实体将所述OAuth程序路由到相应的管理域，所述管理域是访问授权的最终发放者和控制者，并且，所述管理域为所述最终用户提供直接或代理路由，以便访问所述API。

背景技术

[0002] 在过去的几年里，互联网世界已经经历了网络API/网络服务的爆炸，使服务提供商将功能开放给其它网站，并且在很多情况下开放给个人开发者，从而使他们能够迅速构建新的服务或通过包含和结合远程公开的功能来丰富已有的服务。现今，这是互联网中的主要趋势之一，并且预计它将继续增长和发展，逐步地处理更加复杂和动态的场景。

[0003] 开放的API意味着：必须从技术的角度来妥善地解决不同的问题，以得到真正有用且适合的结果。要解决的主要问题之一是安全，并且特别是最终用户隐私的安全。基本上，互联网应用应当只在正确认证并收到来自资源的所有者的明确的访问许可或授权后，才能访问服务提供商的功能而不需要资源所有者(通常是最终用户)共享他/她的身份或应用的证书。作为这个问题的解决方案，在过去几年中出现了新的OAuth标准，目前的互联网虚拟社区和IETF都在这个标准的第二个版本下运行。

[0004] OAuth提供所有的细节用于令人满意的写出在实际访问这些API之前应用已知的所有API端点的静态场景地址，以及所有API从属于的相同的服务提供商/管理域的静态场景地址，并且因此可以由单独的集中点授权和控制所述的访问。然而，由于在互联网上API数量的增长，有新的由不同的服务提供商公开相同的API的场景，或相同的API被相同的服务提供者从多个位置简单地公开，目的是改进所述API公开策略的可扩展性和性能。

[0005] 目前主要的现有技术是由IETF标准化的OAuth1.0协议[1]，和在IETF标准化进程中的OAuth2.0协议标准[2]。

[0006] OAuth1.0和2.0都定义了开放协议，所述协议允许来自桌面和web应用的以简单和标准的方式安全API授权，所述授权对信任和非信任使用者(客户端)都有效。OAuth的规定，直接适用于准许访问REST服务中的资源，但也可以用于例如基于SOAP的Web服务。

[0007] 为了访问资源，客户端首先通过OAuth协议从资源所有者那里获得许可。该许可的表达形式是令牌和可选的相匹配的共享密钥。如已述的，令牌的目的是使资源所有者不必与客户端共享证书。与资源所有者的证书不同，令牌能够以受限的范围和有限的时效发放，并能够单独地撤销。

[0008] 总之，OAuth协议的主要目的是为使用者(consumer)提供用来获得有效的访问令牌的方法。

[0009] 如将会在图1中显示的，有两个起关键作用的令牌：

[0010] - 首先，请求令牌被用作委托授权程序内的一个引用。更具体地说，根据 [1]，使用者使用请求令牌向用户请求授权访问受保护的资源。然后，建议为有限时效的用户授权的请求令牌交换为访问令牌。

[0011] - 最后，使用者使用这个访问令牌代表用户访问所述 API，而不是使用用户的证书(用户名和密码)。访问令牌可能会限制为访问某几个 API 或甚至是一个特定的 API 内的资源。

[0012] 在 OAuth2.0 中，请求令牌的概念消失了。因此，该流程以对服务的授权请求开始。如将在图 2 中示出的，在此请求中发送使用者的身份证明，而不是包括请求令牌。有几种 OAuth2.0 场景 ：

[0013] - 授权码，如将在图 2 中示出。

[0014] - 隐含授权 ：隐含授权类型适合于客户端不能维持客户证书保密(为用授权服务器认证)，例如驻留在用户代理中的客户端应用(如，浏览器)。

[0015] - 资源所有者密码证书 ：资源所有者密码证书授权类型适合于资源所有者与客户拥有信任关系的情况，如计算机操作系统或高度专享的应用。

[0016] - 客户端证书 ：当客户端请求访问的受保护资源在其控制下，或另一个资源所有者已事先与授权服务器商定时(所述方法在 OAuth2.0 规范的范围之外)，客户端可以仅使用其客户端证书请求访问令牌。

[0017] 基础 OAuth 是一种广泛使用的标准并且有许多的实现，如 BlueVia[4]，Twitter[5]，谷歌 [6] 或雅虎 [7] 所提供的。OAuth2.0 仍然是一个草案，但有许多的实现已可用或计划提供 OAuth2.0。实例有 Facebook[8] 和谷歌 [9]。现有的实现对如下事实作出警告 ：OAuth2.0 仍然是一个草案，并因此直到其最终标准化都可能有变化。

[0018] 也有一些解决方案涵盖了集中 OAuth 服务给服务提供商提供用于授权访问受保护资源的 OAuth 的场景。谷歌或雅虎就是这样的情况，其提供的 OAuth 被用于互联网中的任何服务。

[0019] 此外，覆盖相同的场景，谷歌还提供了一个解决方案，由 OAuth 和 OpenID[10]、[11] 结合。有了这种解决方案，谷歌允许第三方网站和应用的访客使用其用户的谷歌帐户登录。他们解释为，这个扩展对使用 OpenID 和 OAuth 的 Web 开发人员是非常有用的，尤其是简化了用户的流程，其仅需依照请求批准一次，而不是两次。

[0020] 不同于标准的 OAuth，OpenID OAuth 扩展在服务器到服务器的请求中，并不提供从联合使用者到在联合供应商处的请求令牌端点的请求令牌。取而代之地，作为所述 OpenID 身份认证响应的一部分联合供应商给联合使用者返回一个已经批准的请求令牌。

[0021] 然后，依照标准 OAuth 做法，联合使用者在联合提供商访问令牌端点处将请求令牌交换为访问令牌。

[0022] 所有这些解决方案的公共点是，他们覆盖 ：有一个全部服务提供商共享的集中 OAuth 服务的方案。也就是所述集中式的解决方案负责发放访问令牌。

[0023] 另外，存在几种解决方案，其中 ’ 令牌 ’ 的概念用于路由的目的。例如，思科和微软 [12] 有个解决方案，该解决方案使用路由令牌信息将客户端会话重定向到微软终端服务器。另外的例子是微软为连接代理 [13] 而使用的令牌重定向，或 TransNexus 为 NexTransitTM 信令代理 [14] 的重定向解决方案。

[0024] 然而，这些解决方案都与 OAuth 协议无关，即：用于路由的令牌不是 OAuth 访问令牌。

[0025] 如之前的评述，就 OAuth 的现有机制而言，有包括两种可能的场景的解决方案：

[0026] - 一个提供受保护资源的服务提供商实现用以提供对受保护资源的访问的 OAuth。服务提供商自己提供所述 OAuth 服务。此场景与常规的 OAuth 相匹配。

[0027] - 一个 OAuth 的服务为 N 个服务提供商提供 OAuth，所述服务提供商提供其拥有的受保护资源。认证和令牌发放由集中 OAuth 服务来完成。因此，用户证书是通过 OAuth 的集中服务提供的集中证书。例如，由雅虎和谷歌提供的服务就是这种。

[0028] 此外，谷歌提供的 OAuth 与 OpenID 的组合，是第二场景的更先进更灵活的使用。

[0029] 总之，在现有的解决方案中，要么是服务提供商提供授权端点，或者是有一个集中授权端点为多个服务提供商进行认证并发放访问令牌。

发明内容

[0030] 需要依赖于一种分布式的场景，该场景中应用事先不知道 API 端点，并且没有用于发放授权和控制其后续使用的集中控制器的单一的实体。考虑到该要求，本文提出的发明目的是：针对 OAuth 高度分布的应用，以及有潜在变化的 API 公开的场景，提供不以任何方式修改 OAuth 标准的详细解决方案。为做到这一点，本发明专注于在 API 公开基础设施本身内部如何解决该问题，而不是修改接口。

[0031] 本发明所涵盖的下列场景，是现有的解决方案没能正确解决的：

[0032] - 若干个服务提供商提供相同的受保护的资源（即 API）。这种情况通常发生在电信（运营商）环境中，其中若干个运营商提供一个受保护的资源，如 SMS 操作。

[0033] - 由于受保护的资源是相同的，要访问受保护资源的应用实现了 OAuth，并期望一个实现将能为每个服务提供商工作。所述应用期望实现上述功能，即使其不知道最终用户是属于某一个服务提供商还是属于其它服务提供商。

[0034] - 关键在于，所有服务提供商可以提供一个集中共享 OAuth 服务，但每个服务提供商需要负责用户身份认证和访问令牌的发放。属于特定服务提供商的用户和用户证书不能被集中 OAuth 服务获知。

[0035] 本发明提供了一种方法，该方法用于提供对服务应用的授权访问，为了使用最终用户的受保护的资源，所述受保护资源通过多个管理域的端点公布，在所述多个管理域中的各个管理域分别负责发放和控制授权访问多个受保护的资源的后续使用，如由属于所述管理域的最终用户授权，与已知的提议不同的是，所述端点是所述服务应用事先不知道的，并且没有集中的授权者 / 控制者，并且发明所述方法的特征在于还包括：

[0036] i. 使用中间实体或全局实体，用于：

[0037] a) 基于至少包括所述用户身份的灵活规则，从所述多个管理域中选择一个管理域；以及

[0038] b) 所述选定的管理域执行安全授权，允许所述最终用户通过开放协议访问；

[0039] ii. 一旦执行了所述安全授权，所述选定的管理域对所述服务应用提供：由所述服务应用，直接或代理访问包括在其中的端点和所述的包含在端点中的受保护的资源。

[0040] 根据所附的权利要求 2 至 19，以及与若干实施例的详细描述有关的随后的章节，

描述了本发明的第一方面的方法的其它实施例。

附图说明

[0041] 参照附图，从下面实施例的详细描述中，可以更充分地理解发明的前述优点和其它优点，这些必须被视为说明性和非限定性的方式，其中：

[0042] 图 1 示出了当前按照 OAuth1.0 的交互。

[0043] 图 2 示出了当前按照 OAuth2.0 的交互。

[0044] 图 3 示出了高度分布和有潜在变化的 API 公开场景的图形表示。

[0045] 图 4 示出了根据本发明的一个实施例的 OAuth 程序的路由。

[0046] 图 5 示出了根据本发明的一个实施例的 API 请求的路由。

[0047] 图 6 示出了根据本发明的一个实施例，当请求令牌能够直接由全局服务基础设施设置，并且所有管理域共享公共的门户网站时，由 OAuth1.0 支持获得访问令牌的时序。

[0048] 图 7 示出了根据本发明的一个实施例，当请求令牌按要求由本地服务基础设施设置产生，并且所有管理域共享公共的门户网站时，由 OAuth1.0 支持获得访问令牌的时序。

[0049] 图 8 示出根据本发明的一个实施例的由 OAuth2.0 支持获得访问令牌的时序，其无需请求令牌，并且所有管理域共享公共的门户网站。

[0050] 图 9 根据本发明的一个实施例，当请求令牌可以直接由全局服务基础设施设置，并且所有管理域都拥有各自的本地门户网站时，由 OAuth1.0 支持获得访问令牌的时序。

[0051] 图 10 示出了根据本发明的一个实施例，当请求令牌按要求由本地服务基础设施设置产生，并且所有管理域都拥有各自的本地门户网站时，由 OAuth1.0 支持获得访问令牌的时序。

[0052] 图 11 根据本发明的一个实施例的由 OAuth2.0 支持获得访问令牌的时序，其无需请求令牌，并且所有管理域都拥有各的自本地门户网站。

[0053] 图 12 示出了根据本发明的一个实施例，在代理模式和重定向模式中使用 (consuming) 受保护资源时的时序图。

[0054] 图 13 示出根据本发明的一个实施例在撤销访问令牌时的时序图。

具体实施方式

[0055] 本发明提供了一种解决方案，用于将 OAuth1.0 和 / 或 2.0 应用到高度分布和有潜在变化的 API 公布的场景。如图 3 所示，根据该场景的图形化表示，必须考虑到下列需求：

[0056] -API 端点 (即，公开的服务或资源) 兼容 OAuth1.0 或 OAuth2.0，并且对彼此或任何全局基础设施没有任何了解。

[0057] - 每个 API 端点属于一个管理域，该管理域负责对本地的 API 端点实施访问控制。管理域通常对应于服务提供商，但一个服务提供商可能有多个管理域。

[0058] - 为了每个最终用户必须使用的外部应用对 API 端点或管理域不具有任何预先了解。

[0059] -API 公开的点和管理域能够随着时间改变。

[0060] 首先，由于应用对其必须联系的端点和管理域不具有任何预先了解，需要有静态的 URL 使应用能够初始触发 OAuth 的过程，因此 OAuth 协议端点 (URL) 一直是为了获得访问

任何底层 API 端点的授权的唯一和合法的端点。此外，在应用必须进行身份认证的过程中，必须有一个支持应用注册和证书管理的“全局基础设施”。它提供了一种可以理解的机制，该机制适合用于外部使用者自行注册和触发 API 访问，无需处理底层的分布式体系结构的细节。全局基础设施知晓这些细节并且对外部应用隐藏这些细节。描述全局基础设施如何知道关于本地管理域和 / 或 API 端点已超出了本发明的范围，但在原则上有不同的选择(静态配置，动态注册过程等)。总之注意所述全局基础设施不被用作后续 API 使用控制的集中媒介是非常重要的，它只是用于支持以下描述的方案的两个主要概念 ：

[0061] - 路由 OAuth 过程 ：没有必要集中控制和执行访问授权，取而代之的是必须直接处理授权并由 API 所属的本地管理域强制执行。这使得本地的基础设施的改变不会影响整个系统，并且，为每个域使用自己的访问策略而不必强制遵从全局安全策略提供了灵活性。为了做到这一点，必须将 OAuth 过程以灵活的规则路由到基于资源所有者的在 OAuth 的授权阶段提供的信息和首选项的适当管理域。

[0062] 从一个场景到另一个场景，路由的标准可能会有所不同。例如，明显的路由决策可能会将 OAuth 过程路由到用户所属的服务提供商，但该解决方案并不排除其它标准最适用于目标场景。关键点是，将 OAuth 过程灵活的路由到基于资源所有者的信息和首选项的适合的本地管理域。

[0063] - 路由 API 请求 ：一旦正确的管理域(更具体地，该域的授权服务器) 已经发出授权，并已取得与该应用相关联的访问令牌，由应用发送的 API 请求能够基于该访问令牌路由。事实上，除了在授权阶段作为标识符或指向获得访问范围和策略的指针之外，访问令牌在授权阶段其本身也作为指向的路由决策的指针。这样一来，知道访问令牌作为 OAuth 过程的媒介的全局服务基础设施提供数据库，该数据库将每个访问令牌与适合的管理域联合起来，并且可选地所述特定的 API 端点可以在所述管理域中以所述令牌访问。

[0064] 基于这个概念，如在图 5 中所示，在下述两种方式之一中，对所述 API 端点的访问可以是分布式的，不依赖于任何集中元素 ：

[0065] - 重定向模式 ：在发送 API 请求之前，应用询问用于发放访问令牌的端点。应答可以被缓存在所述应用侧，于是后续的查询就不再需要重复所述请求。

[0066] - 代理模式 ：一组“API 代理”在互联网上可以是分布式的。应用向最近的代理发送 API 请求而不必操心实际要使用的 API 端点。当请求到达代理时，该代理提取访问令牌，并通过询问所述全局设施，获得与所述访问令牌相关的 API 端点。一旦代理获知所述端点，它就将请求简单地转发到该端点。与所述访问令牌关联的端点可以由代理缓存从而避免之后请求使用同样的令牌时再次询问。所述应用如何发现所述最近代理，已经超出了本发明的范围。

[0067] 分布式 OAuth 提供了简单而有效的方法，来授权和代表用户使用服务，其中的新颖性在于针对授权以及资源服务器(即 API 端点)由路由决策动态决定，例如，针对用户的服务提供商(更确切地说，是服务提供商的适合的管理域)。为了简单起见，在下面的详细描述中，假设这是始终应用的路由决策，但正如之前所指出的，如果认为适合特定的场景，有应用其它的路由标准的自由。

[0068] 为了以可理解的方式来描述所提出发明的细节，依照如下顺序 ：

[0069] - 首先，对这些在真实情况下首先发生的过程，即获得授权，提供描述。获得授权的

过程被进一步分为两个不同的选择方案：

[0070] 1、所有的服务提供商 / 管理域，使用一个公共的、全局的认证和授权门户来与最终用户交互的情况(请注意，授权发放及后续的控制仍然是由每个管理域独立控制)。

[0071] 2、每个服务提供商 / 管理域拥有自己的本地认证和授权门户来与最终用户交互的情况。

[0072] - 其次，描述允许路由 API 请求的过程，该 API 请求的路由基于完成授权阶段之后获得的访问令牌。

[0073] 此外，对每个场景考虑两个平行的 OAuth1.0 的解决方案：一个方案假设请求令牌可以直接指向本地的服务基础设施，以及另一个方案否定以上假设，并且按需要由本地服务基础设施动态生成请求令牌，依照标准的 OAuth1.0 接口，此方案是完全本地公布。

[0074] • 一个全局门户网站

[0075] 在使用受保护资源之前，应用需要授权它代表用户访问该资源的访问令牌。本节考虑一个全局门户网站与最终用户进行交互以执行认证和授权，而不考虑底层授权服务器或与用户相关联的资源服务器。这个门户网站可以帮助为每个授权和资源服务器都提供统一的用户体验。但是，要注意到所述身份认证和授权实际上是由本地服务基础设施执行。

[0076] 如图 6 所示，该过程分为以下的步骤：

[0077] 1、用户与应用交互。应用需要用户的授权，以访问某些受保护的资源。

[0078] 2、应用从所述全局授权服务器获取请求令牌，因为应用不清楚与用户相关的本地服务基础设施。

[0079] 3、将用户与请求令牌一起重定向到全局门户网站，以便对应用进行认证和授权。由全局基础设施确认请求认证令牌，并且，所述门户网站获得应用的标识符和范围(应用所要求哪些资源)。

[0080] 4、用户在全局门户网站进行身份认证。用户名使得全局基础设施能找出与用户相关的本地服务基础设施(通过一个将用户名映射到与相关的本地服务基础设施的数据库。如前面所述，描述如何使得该本地信息能为所述全局基础设施所用超出了本发明的范围，但也有不同的可选方案，如使用订阅的方法，数据复制解决方案等)。所述认证请求被传送到与用户相关联的本地服务的基础设施。

[0081] 5、一组交互可能发生在全局基础设施和本地服务的基础设施之间，用于建立授权的条款及条件。这些交互超出了本发明的范围。

[0082] 6、用户在特定条款及条件下授权应用代表他 / 她访问一些受保护的资源。请求令牌存储在本地服务的基础设施中，用于使所述操作遵循标准的 OAuth 过程。本地实例生成 OAuth1.0 术语中的授权码、认证码，该授权码、认证码被缓存在全局的基础设施用于将授权码映射到本地服务基础设施。要注意，这一步实际上是一组对本地和全局基础设施的要求，但它并不影响 OAuth 的公共 API，因为 getRequestToken 实际上是适用于全局的，如在目前场景的第 2 步所描述的，即外部应用总是看到一个标准的 OAuth 访问程序。

[0083] 7、应用收到的授权码(或认证码)通过 HTTP 重定向(其它的机制也将是可行的)。应用向全局服务基础设施请求访问令牌，所述全局服务基础设施是实际生成访问令牌(和令牌密钥)的本地服务基础设施中的授权服务器的中间媒介。全局基础设施保存访问令牌和本地服务基础设施之间的映射，所述本地服务基础设施使得全局的访问令牌目录能将所

述请求路由到资源。最后，应用获得访问令牌(和令牌密钥)，以便代表用户使用受保护的资源。

[0084] 在图 7 中，对按需求由本地基础设施生成请求令牌的情况下的上述过程进行了描述。以下列方式形成在 CallBack2 中的流 :AAPortalURL?rt1=RequestToken1&cb1=callBack1。这允许 AA 门户网站自动恢复 RequestToken1 和 callback1，它们将在返回到所述曾经发出授权的应用时使用。

[0085] OAuth 的 2.0 版本简化了整个过程，因为应用不再需要请求令牌。在用于身份认证的 HTTP 重定向步骤中，该应用还包括一个应用范围，用于指定用户将授权的资源，如图 8 中所示。

[0086] • 分布式本地门户网站

[0087] 在这种情况下，有没有全局的门户网站与最终用户实施交互，以便执行身份认证和授权。每个本地服务基础设施提供了一个门户网站，以执行这两个功能。认证和授权都不通过全局服务基础设施作为中间媒介。其结果是，需要一种替代的方法，用于针对相应的本地服务基础设施。可能会用到几种机制(例如，通过分析用户的 IP 地址)来推断本地服务的基础设施。这种机制可能通过请求用户明确地选择来进行简化，如图 9 中所示，它描述了获取访问令牌的时序，假设请求令牌可以直接本地设置。此过程分为以下步骤 :

[0088] 1、用户与应用交互。应用需要用户的授权以便访问一些受保护的资源。

[0089] 2、应用从的全局授权服务器获取请求令牌，因为所述应用不清楚与用户相关的本地服务基础设施。

[0090] 3、将用户与请求令牌一起重定向到全局门户网站。由全局基础设施认证请求令牌，并且该门户网站获取了应用标识符和范围(应用所要求哪些资源)。

[0091] 4、用户选择本地服务基础设施(通过点击网页或超出本发明范围的替代机制)。通过 HTTP 将用户重定向到本地门户网站(根据所述选择)，并且为了将授权码映射到本地服务基础设施，所述应用的回调函数被替换为全局服务基础设施的 URL。

[0092] 5、用户在本地门户网站的进行身份认证。

[0093] 6、为设立所述条款及条件，可能会发生一组交互。这些交互超出了本文的讨论范围。

[0094] 7、用户在特定条款及条件下授权应用代表他 / 她访问一些受保护的资源。请求令牌存储在本地服务的基础设施上，使所述操作遵循标准的 OAuth 过程，但是，所述令牌密钥将是一个预先建立的值，因为它不能附加在用户的请求中。最后，本地实例生成 OAuth1.0 术语中的授权码、认证码。要注意，这一步实际上是一组对本地和全局基础设施的要求，但它并不影响 OAuth 的公共 API，因为 getRequestToken 实际上是适用于全局的，如在目前场景的第 2 步所描述的，即外部应用总是看到一个标准的 OAuth 访问程序。

[0095] 8、将用户与请求令牌和授权码一起重定向到全局服务基础设施，使得授权码和本地服务的基础设施之间的映射缓存到全局基础设施。应用的回调是从应用的详细信息获得的。用户被重定向到应用的回调。

[0096] 9、应用收到的授权码(或 OAuth1.0 中的认证码) 通过 HTTP 重定向(其它的机制也将是可行的)。应用向全局服务基础设施请求访问令牌，所述全局服务基础设施是实际上生成访问令牌(和令牌密钥)的本地服务基础设施中的授权服务器的中间媒介。全局基础设施

保存访问令牌和本地服务基础设施之间的映射，并且所述本地服务基础设施使全局的访问令牌目录能将所述请求路由到资源。最后，应用获得访问令牌(和令牌密钥)，以便代表用户使用受保护的资源。

[0097] 在图 10 中，对按需求由本地基础设施生成请求令牌的情况下的上述过程进行了描述。以下列方式形成在 CallBack2 中的流 :GlobalPortalURL?rt1=RequestToken1&cb1=callBack1。这允许全局门户网站自动恢复 RequestToken1 和 callback1，它将在返回到所述曾经发出授权的应用时使用。

[0098] 在 OAuth2.0 中，如图 11 中所示，如果没有包含在应用中的所述状态包函在全局门户中。它用于在用户授权后关联从本地门户网站返回的第二个重定向。

[0099] • 使用受保护的资源

[0100] 一旦应用获得由如上所述的用户服务的提供商发放的为了访问受保护的资源的访问令牌，应用的服务请求需要针对用户的资源服务器。虽然应用不清楚关联到用户的是哪个资源服务器，用以下可选方案所述访问令牌将帮助应用找到关联的资源服务器 :

[0101] - 自包含。用户的资源服务器在访问令牌中是自包含的。在这种情况下，有两种另外的可能性 :

[0102] 1、资源服务器的信息被加密，以防止应用(第三方)从访问令牌推断信息。如果用户的资源服务器被视为敏感信息，这种方式是有益的。

[0103] 2、资源服务器信息可以明文获取，其简化了访问令牌处理。

[0104] - 基于目录。访问令牌的目录将每个访问令牌与资源服务器相关联。

[0105] 其结果是，到适当的资源服务器的路由是基于访问令牌的。考虑两种不同的方法 :

[0106] - 代理模式。代理将介于应用和服务器资源之间。所述代理负责基于访问令牌将应用的服务请求路由到用户的资源服务器。

[0107] - 重定向模式。应用从访问令牌的目录中检索出资源服务器端点。然后应用必须构造访问在资源服务器中受保护资源的 URL。

[0108] 代理模式有两个优点 :a)URL 是唯一的 ;在每个资源服务器中相同的 URL 访问相同的受保护资源，以及 b)所述应用被简化，因为它不知道重定向关系。另一方面，此模式的主要缺点是中间媒介增加了响应延迟并可能成为瓶颈，这可以通过复制和地理上分布多个代理来缓解。如图 12 中所示，代理模式被分成两种情况 :

[0109] - 应用以无效访问令牌访问。通过全局访问令牌的目录，代理不成功地尝试解析所述访问令牌。所述代理将以未经授权的 HTTP 消息回复。

[0110] - 该应用使用一个有效的访问令牌。代理试图用缓存来解析访问令牌，以加快路由。如果访问令牌没有存储在缓存中，该代理试图通过全局访问令牌目录解析它，该全局访问令牌目录将返回适合的资源服务器，并为将来的请求将其暂时保存在缓存中。最后，代理访问将所述资源。

[0111] 所述重定向模式是最有效的方法，因为应用与相应的资源服务器直接进行交互。但是，应用逻辑稍微复杂一些，因为它需要通过全局访问令牌目录解析访问令牌，如图 12 中所示。为进一步使用，该应用将这个映射缓存在本地。

[0112] 如果所述资源服务器端点在访问令牌中自包含，访问令牌的解析可以在全局的访

问令牌目录或直接在代理服务器 / 应用中执行。

[0113] • 撤销访问令牌

[0114] 用户需要撤销对应用的授权的机制，这意味着撤销访问令牌。本地服务基础设施由于具体策略也可能撤销访问令牌(例如，一个访问令牌只能使用一次，一旦应用使用了它)。

[0115] 如图 13 中所示，撤销访问令牌的过程，包括如下步骤 ：

[0116] 1、用户进入到全局门户网站以撤销对应用的授权。该门户网站将这个撤销传递到全局服务的基础设施，该设施为适当的本地服务基础设施的中间媒介。

[0117] 2、本地服务的基础设施将撤销由本地的授权服务器管理的访问令牌。所述全局服务基础设施还将撤销的访问令牌从其缓存中删除。

[0118] 自此，也可以直接向本地的服务基础设施发起撤销，并且令牌可能没有被撤销就到期了，为了在全局层面上获得有效的访问令牌的最新信息，全局服务基础设施定期从每一个本地服务的基础设施检索撤销访问的令牌，以便更新访问令牌缓存列表。这种同步的没有严格的时间要求，是因为用无效的访问令牌没有方法访问资源，同步仅是从全局的服务基础设施的缓存中消除了无效的实体记录。

[0119] 电信运营商市场是本发明的主要焦点。每个电信运营商成为服务提供商，并且每个用户(或客户)属于某个运营商。通常，应用市场可能提供构建在电信运营商的服务上的应用。然而，应用开发人员需要实现相同的应用的多个变体，以便适用于多个电信运营商，除非每个运营商所提供的服务接口是标准化的。

[0120] 标准化的服务接口将使应用开发者受益，因为它拓宽了潜在用户和电信运营商的市场，所述用户随着应用数量增长和质量变好而增加，并且，所述电信运营商因更高的交易量成为他们收入的增长点。使用例中，所有该用例的参与者都从这种方法获利。

[0121] 服务提供商(电信运营商)信任高级的、全局实体来进行身份认证和授权或为上述目的提供一个本地的门户网站。如果用户像信任它们的电信运营商一样对这个全局实体有信心，那么前一种情况是有效的。

[0122] 然而，其他用例不能基于该全局实体。大规模的应用社区(WAC)[15] 旨在建立对每个运营商和手机都有效的应用市场。然而，运营商不能对不同的实体开放他们的对业务至关重要的用户信息。后一种方法中在本地进行身份认证和授权，将有助于解决这一约束，并且用户将受益于广阔的应用设置，这些应用使用电信服务(为了这些已经标准化的服务)。

[0123] • 本发明的优点

[0124] 与传统的方法相比，对最终用户和服务提供商分布式的 OAuth 都显示出许多好处。以下是其中一些突出的好处 ：

[0125] - 可扩展性和性能。分布式的 OAuth 避免了以集中的基础设施来发放和强制使用访问令牌的需要，从而使 API 能在一个完全分布式的环境中访问。这意味着在可伸缩性和性能方面有很大的改善，因为基础设施能够水平增长，以解决互联网规模的服务需求。在保持充分控制 API 访问的同时可避免瓶颈。

[0126] - 促进标准的 API。分布式的 OAuth 为不同的服务提供商提供了所需的灵活性，以便在保持其自主性的同时公开相同的 API。只要他们遵循标准的 OAuth 并遵循在本文中公

开的要求，服务供应商可以自由地实现自己的本地 API 的访问控制机制(用户认证，访问策略等)，即使这些 API 在全局场景下将要与其他的服务提供商公共公开。系统的整体操作是令人满意的，而本地实现不需要知道彼此的任何事。

[0127] -简化开发人员体验。目前，开发人员需要在能够访问它们之前知道 API 端点。这一事实使得上述场景复杂化了，该场景中有几个端点和不同的服务提供商，开发人员需要自己应对这种多样性。

[0128] 分布式的 OAuth 对 OAuth 的 1.0 或 2.0 标准保持完全兼容，并且无需来自应用端的任何特殊的行为。得益于分布式的 OAuth，开发人员将能够访问可能属于不同的服务提供商的分布式的 API，只需符合标准的 OAuth 而不必担心底层的多样性。

[0129] 更确切地说，当使用代理模式时应用不需要做任何特别的事情，如果使用重定向模式，应用将需要使用全局访问令牌目录服务。这是在应用方对“标准”操作的唯一添加。

[0130] 本领域的技术人员在不脱离本发明的如其在所附权利要求中定义的范围内可以引入对所描述的实施例的改变和修改。

[0131] 缩略语

[0132] API 应用程序接口

[0133] HTTP 超文本传输协议

[0134] IETF 互联网工程任务组

[0135] REST 表述性状态转移

[0136] SDP 服务传输平台

[0137] SOAP 简单对象访问协议

[0138] URL 统一资源定位

[0139] 参考文献

[0140] [1]OAuth1.0，http://tools.ietf.org/html/rfc5849

[0141] [2]OAuth2.0，http://tools.ietf.org/html/draft-ietf-oauth-v2-22

[0142] [3]http://www.sitepen.com/blog/2009/02/19/introducing-oauth-in-dojox/

[0143] [4]BlueVia developers OAuth guides，

[0144] https://bluevia.com/en/knowledge/APIs.API-Guides.OAuth

[0145] [5]Twitter developers information，http://dev.twitter.com/pages/auth

[0146] [6]Google information on OAuth，http://code.google.com/p/oauth/

[0147] [7]Yahoo developers oauth information，http://developer.yahoo.com/oauth/

[0148] [8]Facebook developer information

[0149] http://developers.facebook.com/docs/authentication/

[0150] [9]Google information on OAuth2.0，

[0151] http://code.google.com/intl/es-ES/apis/accounts/docs/OAuth2.html

[0152] [10]Federated Login for Google Account Users，

[0153] http://code.google.com/intl/es-ES/apis/accounts/docs/OpenID.html

[0154] [11]OpenID OAuth Extension，

[0155] http://step2.googlecode.com/svn/spec/openid_oauth_extension/latest/

openid_oauth_extension.html

[0156] [12]Microsoft Session Directory Services Token Redirection Support in the Cisco Content SwitchingModule for the Cisco Catalyst6500,

[0157] http://www.cisco.com/en/US/prod/collateral/modules/ps2706/ps780/product_solution_overview0900aecd806fc547.pdf

[0158] [13]About IP Address and Token Redirection,

[0159] http://technet.microsoft.com/en-us/library/cc732852.aspx

[0160] [14]NexTransit TM Interconnect Proxy,

[0161] http://www.transnexus.com/Products/NexTransit.htm

[0162] [15]WAC-Wholesale Application Community,http://www.wacapps.net

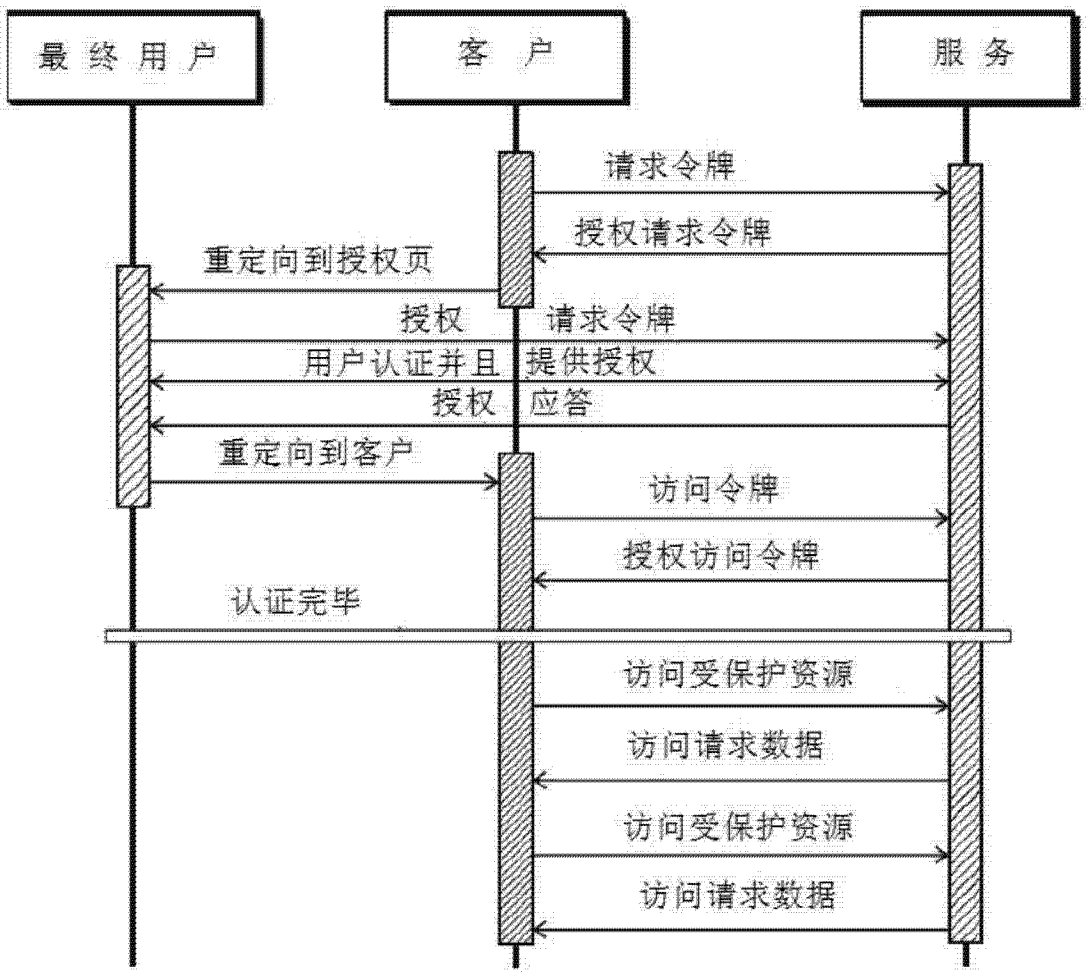

最 终 用 户
客 户
服 务
请求令牌
授权请求令牌
重定向到授权页
授权
请求令牌
用户认证并且
提供授权
授权
应答
重定向到客户
访问令牌
授权访问令牌
认证完毕
访问受保护资源
访问请求数据
访问受保护资源
访问请求数据

图 1

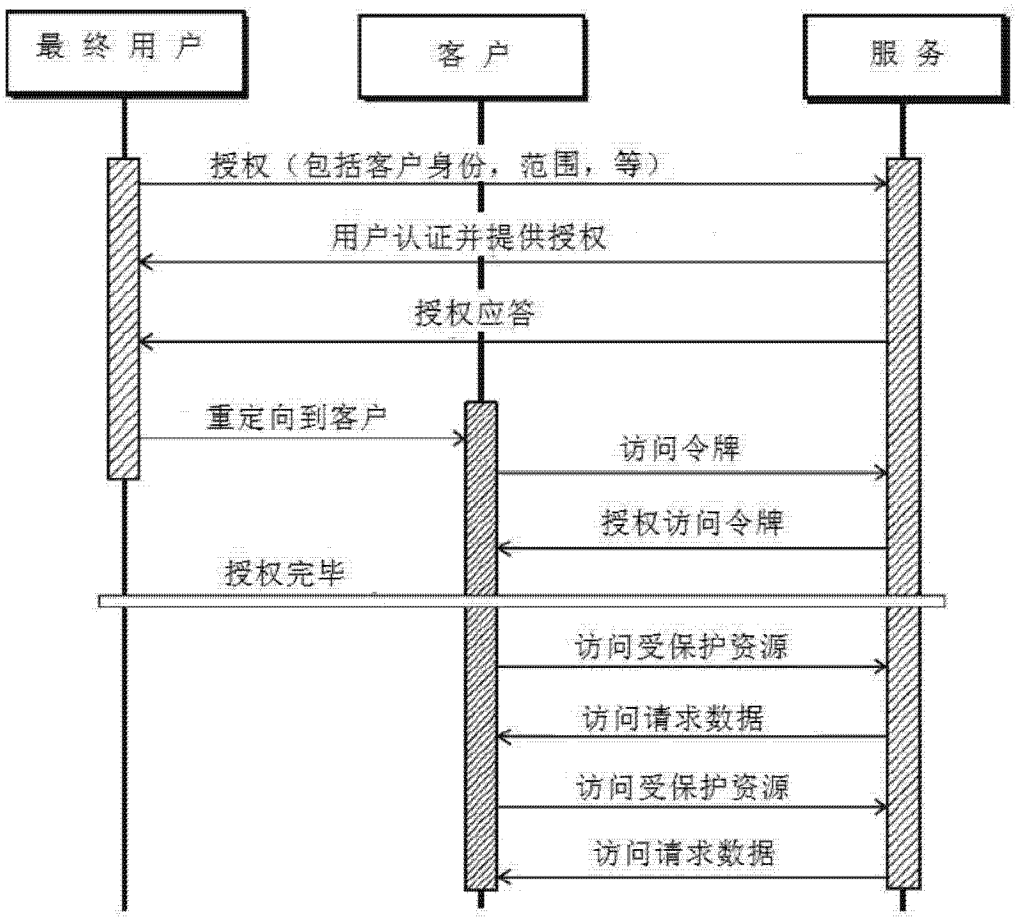

最终用户
客户
服务
授权（包括客户身份，范围，等）
用户认证并提供授权
授权应答
重定向到客户
访问令牌
授权访问令牌
授权完毕
访问受保护资源
访问请求数据
访问受保护资源
访问请求数据

图 2

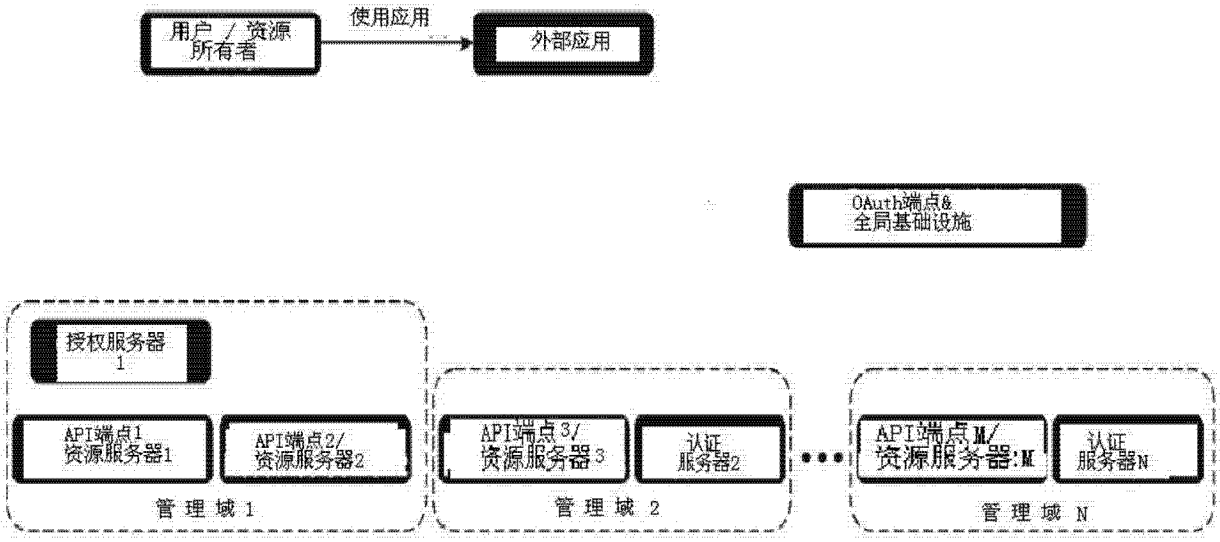

用户 / 资源所有者
使用应用
外部应用
OAuth端点&
全局基础设施
授权服务器 1
API端点1
资源服务器1
API端点2/
资源服务器2
管理域 1
API端点3/
资源服务器3
认证
服务器2
管理域 2
API端点M/
资源服务器M
认证
服务器N
管理域 N

图 3

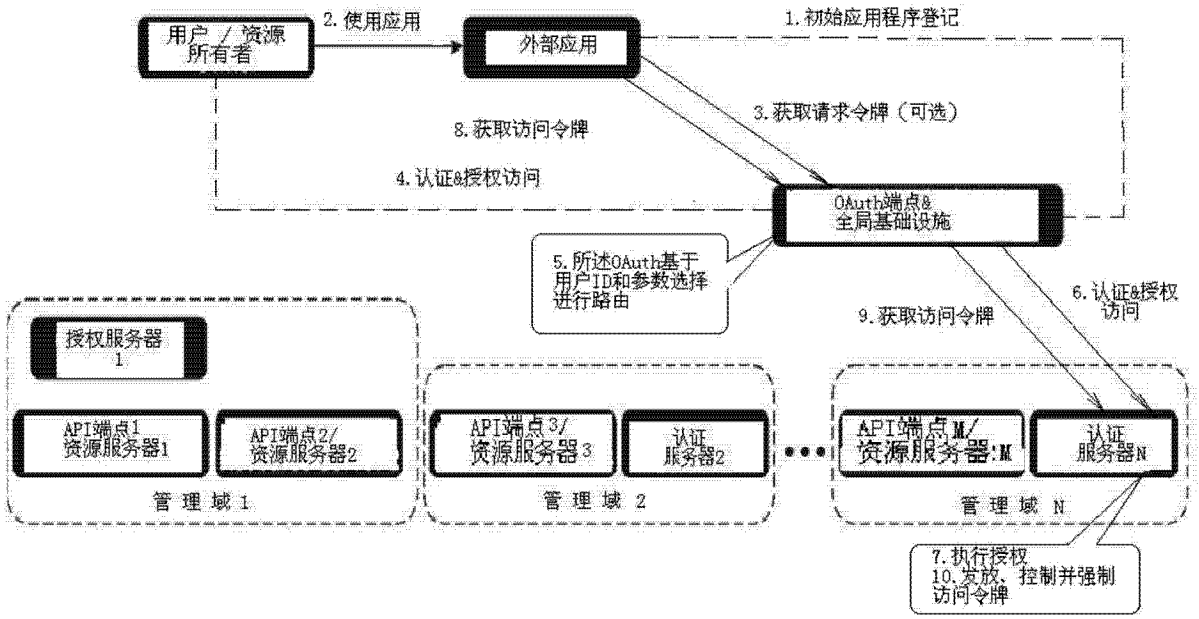

用户 / 资源所有者
2. 使用应用
外部应用
1. 初始应用程序登记
3. 获取请求令牌（可选）
8. 获取访问令牌
4. 认证&授权访问
OAuth端点&全局基础设施
5. 所述OAuth基于用户ID和参数选择进行路由
6. 认证&授权访问
9. 获取访问令牌
授权服务器1
API端点1资源服务器1
API端点2/资源服务器2
API端点3/资源服务器3
认证服务器2
API端点M/资源服务器:M
认证服务器N
管 理 域 1
管 理 域 2
管 理 域 N
7. 执行授权
10. 发放、控制并强制访问令牌

图 4

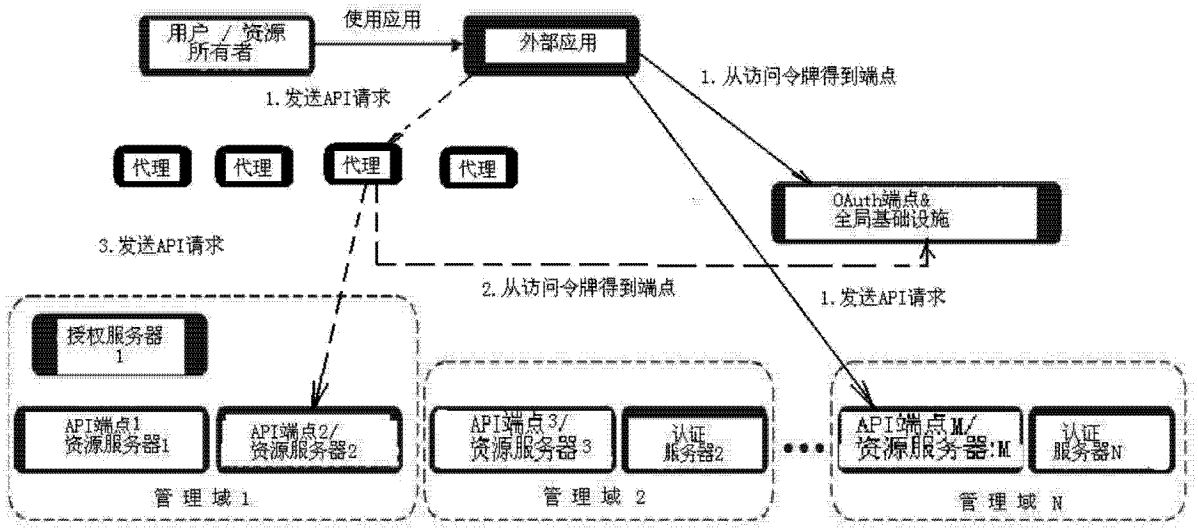

用户 / 资源所有者
使用应用
外部应用
1. 发送API请求
1. 从访问令牌得到端点
代理
代理
代理
代理
OAuth端点&全局基础设施
3. 发送API请求
2. 从访问令牌得到端点
1. 发送API请求
授权服务器1
API端点1资源服务器1
API端点2/资源服务器2
API端点3/资源服务器3
认证服务器2
API端点M/资源服务器:M
认证服务器N
管 理 域 1
管 理 域 2
管 理 域 N

图 5

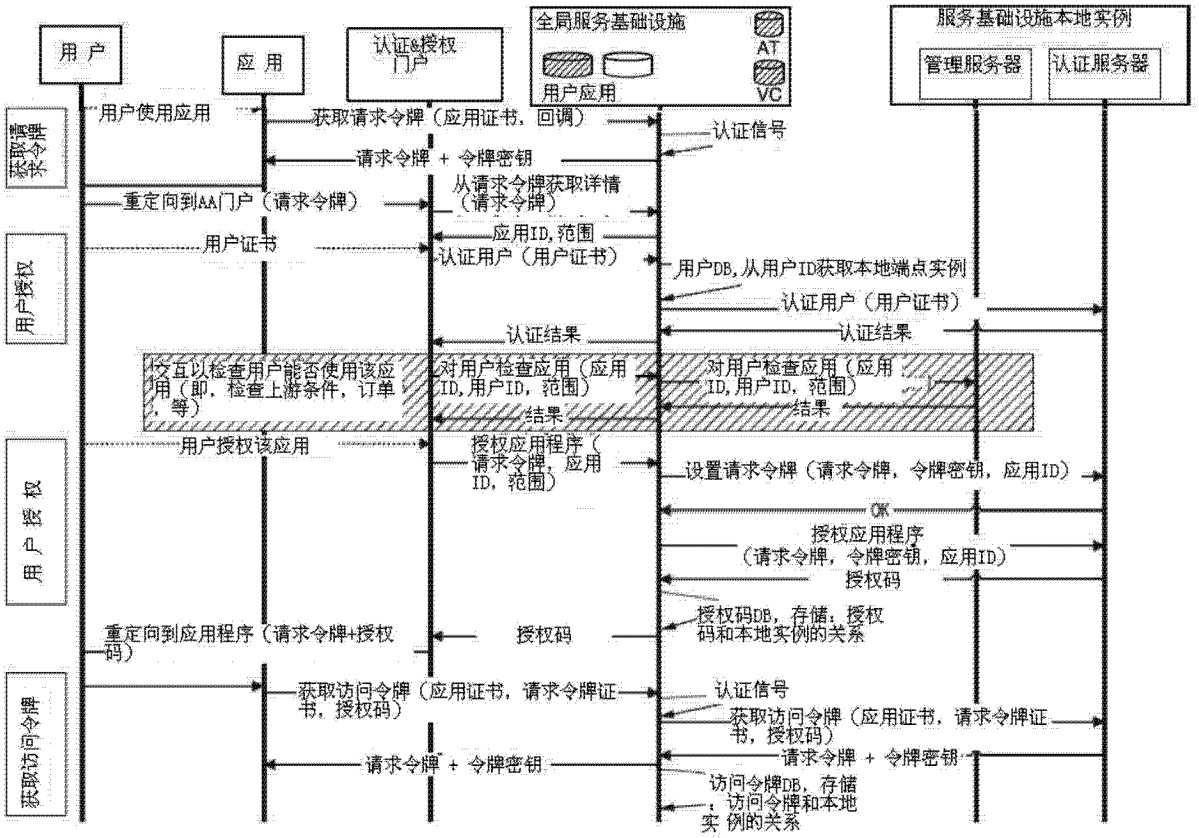

用 户
应 用
认证&授权门户
全局服务基础设施
AT
VC
用户应用
服务基础设施本地实例
管理服务器
认证服务器
获取请求令牌
用户授权
用户授权
获取访问令牌
用户使用应用
获取请求令牌（应用证书，回调）
认证信号
请求令牌 + 令牌密钥
从请求令牌获取详情（请求令牌）
重定向到AA门户（请求令牌）
应用ID,范围
用户证书
认证用户（用户证书）
用户DB,从用户ID获取本地端点实例
认证用户（用户证书）
认证结果
认证结果
交互以检查用户能否使用该应用（即，检查上游条件，订单，等）
对用户检查应用（应用ID,用户ID，范围）
对用户检查应用（应用ID,用户ID，范围）
结果
结果
用户授权该应用
授权应用程序（请求令牌，应用ID，范围）
设置请求令牌（请求令牌，令牌密钥，应用ID）
OK
授权应用程序（请求令牌，令牌密钥，应用ID）
授权码
授权码DB，存储，授权码和本地实例的关系
重定向到应用程序（请求令牌+授权码）
授权码
获取访问令牌（应用证书，请求令牌证书，授权码）
认证信号
获取访问令牌（应用证书，请求令牌证书，授权码）
请求令牌 + 令牌密钥
请求令牌 + 令牌密钥
访问令牌DB，存储，访问令牌和本地实 例的关系

图 6

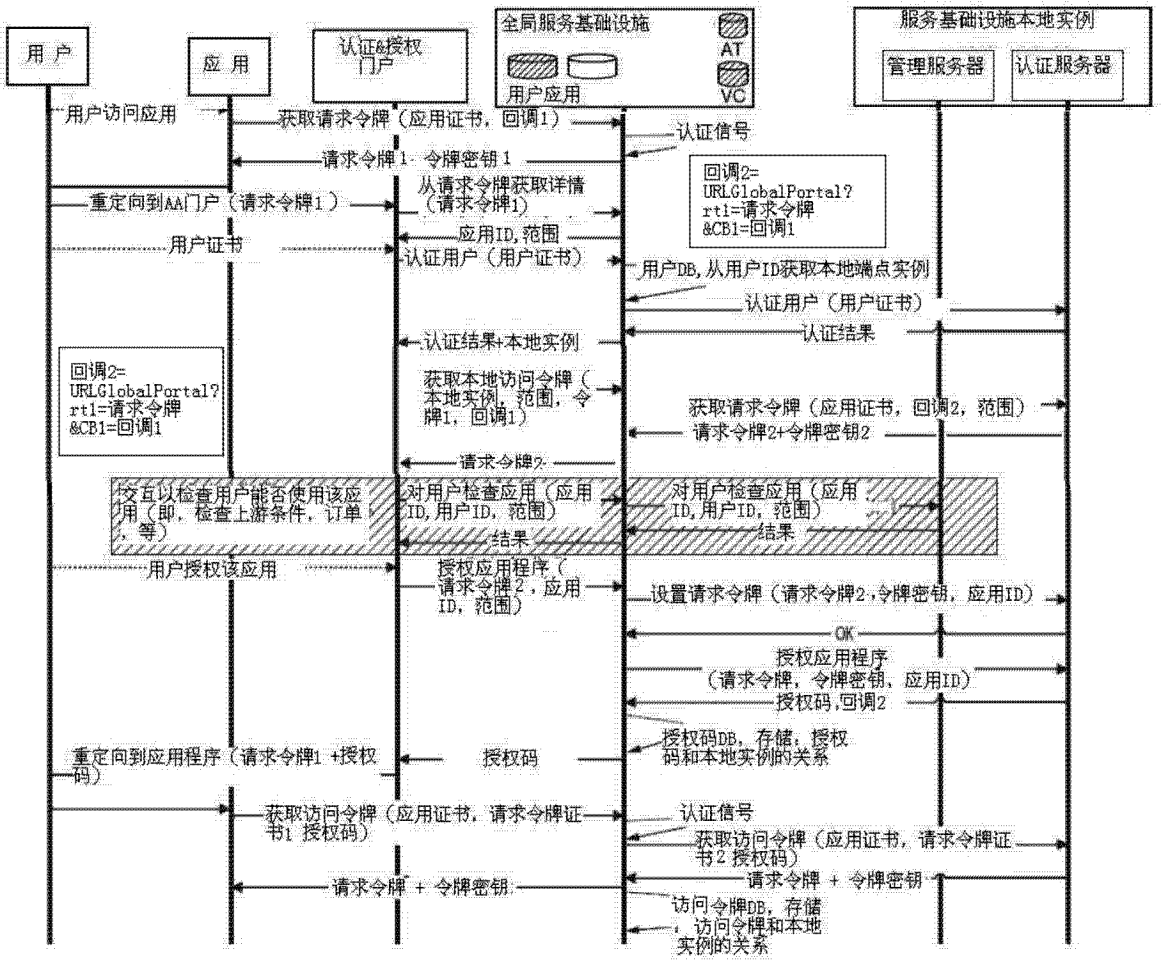

用 户
应 用
认证&授权
门户
全局服务基础设施
AT
VC
用户应用
服务基础设施本地实例
管理服务器
认证服务器
用户访问应用
获取请求令牌（应用证书，回调1）
认证信号
请求令牌1 令牌密钥1
回调2=
URLGlobalPortal?
rtl=请求令牌
&CB1=回调1
重定向到AA门户（请求令牌1）
从请求令牌获取详情
（请求令牌1）
应用ID,范围
用户证书
认证用户（用户证书）
用户DB,从用户ID获取本地端点实例
认证用户（用户证书）
认证结果
认证结果+本地实例
回调2=
URLGlobalPortal?
rtl=请求令牌
&CB1=回调1
获取本地访问令牌（
本地实例，范围，令
牌1，回调1）
获取请求令牌（应用证书，回调2，范围）
请求令牌2+令牌密钥2
请求令牌2
交互以检查用户能否使用该应
用（即，检查上游条件，订单
，等）
对用户检查应用（应用
ID,用户ID，范围）
对用户检查应用（应用
ID,用户ID，范围）
结果
结果
用户授权该应用
授权应用程序（
请求令牌2，应用
ID，范围）
设置请求令牌（请求令牌2,令牌密钥，应用ID）
OK
授权应用程序
（请求令牌，令牌密钥，应用ID）
授权码,回调2
授权码DB，存储，授权
码和本地实例的关系
重定向到应用程序（请求令牌1 +授权
码）
授权码
获取访问令牌（应用证书，请求令牌证
书1 授权码）
认证信号
获取访问令牌（应用证书，请求令牌证
书2 授权码）
请求令牌 + 令牌密钥
请求令牌 + 令牌密钥
访问令牌DB，存储
，访问令牌和本地
实例的关系

图 7

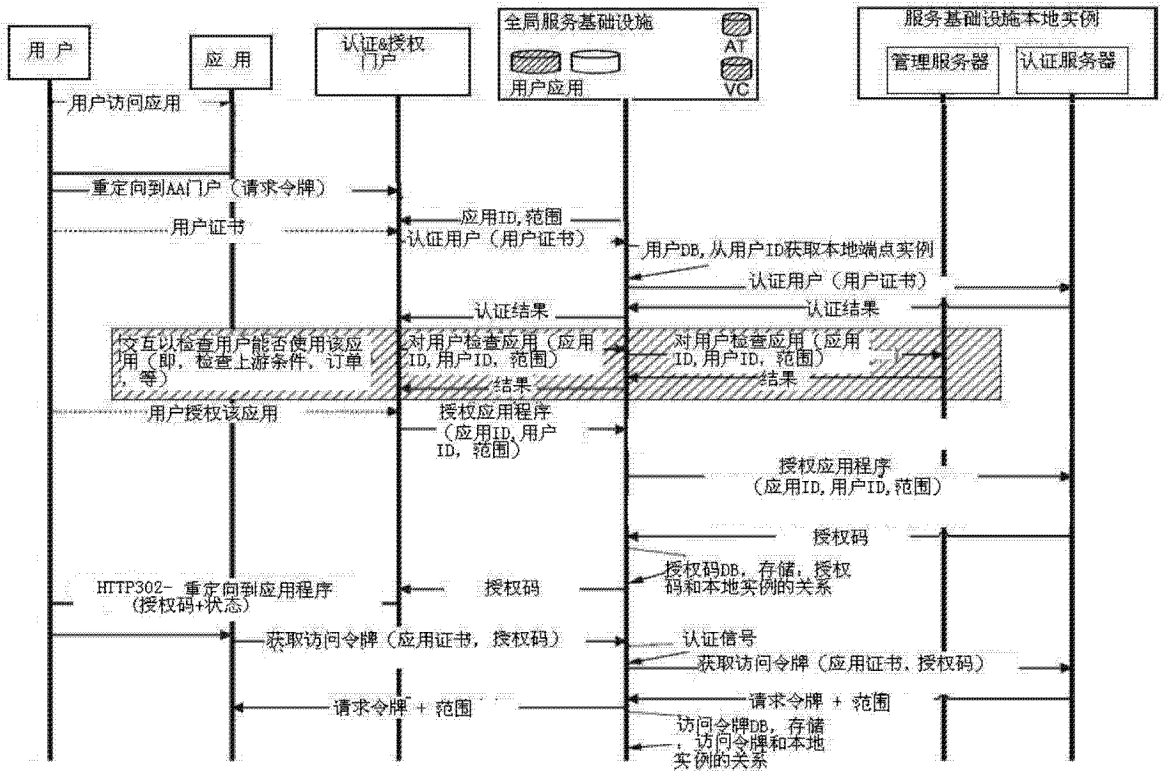

用 户
应 用
认证&授权门户
全局服务基础设施
AT
VC
用户应用
服务基础设施本地实例
管理服务器
认证服务器
用户访问应用
重定向到AA门户（请求令牌）
用户证书
应用ID,范围
认证用户（用户证书）
用户DB,从用户ID获取本地端点实例
认证用户（用户证书）
认证结果
认证结果
交互以检查用户能否使用该应用（即，检查上游条件，订单，等）
对用户检查应用（应用ID,用户ID，范围）
对用户检查应用（应用ID,用户ID，范围）
结果
结果
用户授权该应用
授权应用程序（应用ID,用户ID，范围）
授权应用程序（应用ID,用户ID,范围）
授权码
授权码DB，存储，授权码和本地实例的关系
HTTP302- 重定向到应用程序（授权码+状态）
授权码
获取访问令牌（应用证书，授权码）
认证信号
获取访问令牌（应用证书，授权码）
请求令牌 + 范围
请求令牌 + 范围
访问令牌DB，存储，访问令牌和本地实例的关系

图 8

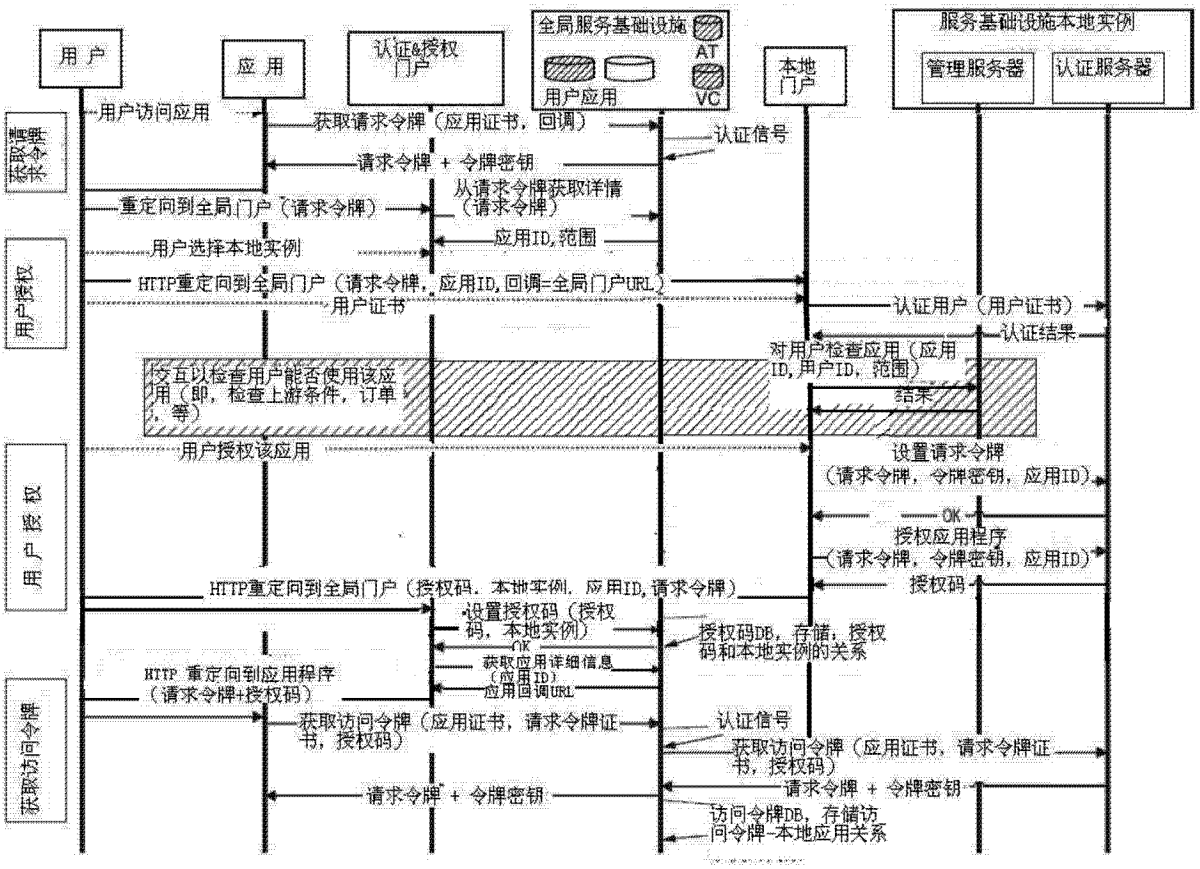

用 户
应 用
认证&授权门户
全局服务基础设施
AT
VC
用户应用
本地门户
服务基础设施本地实例
管理服务器
认证服务器
获取请求令牌
用户授权
用户授权
获取访问令牌
用户访问应用
获取请求令牌（应用证书，回调）
认证信号
请求令牌 + 令牌密钥
重定向到全局门户（请求令牌）
从请求令牌获取详情（请求令牌）
应用ID,范围
用户选择本地实例
HTTP重定向到全局门户（请求令牌，应用ID,回调=全局门户URL）
用户证书
认证用户（用户证书）
认证结果
对用户检查应用（应用ID,用户ID，范围）
结果
交互以检查用户能否使用该应用（即，检查上游条件，订单，等）
用户授权该应用
设置请求令牌（请求令牌，令牌密钥，应用ID）
OK
授权应用程序（请求令牌，令牌密钥，应用ID）
授权码
HTTP重定向到全局门户（授权码，本地实例，应用ID,请求令牌）
设置授权码（授权码，本地实例）
OK
授权码DB，存储，授权码和本地实例的关系
获取应用详细信息（应用ID）
应用回调URL
HTTP 重定向到应用程序（请求令牌+授权码）
获取访问令牌（应用证书，请求令牌证书，授权码）
认证信号
获取访问令牌（应用证书，请求令牌证书，授权码）
请求令牌 + 令牌密钥
请求令牌 + 令牌密钥
访问令牌DB，存储访问令牌-本地应用关系

图 9

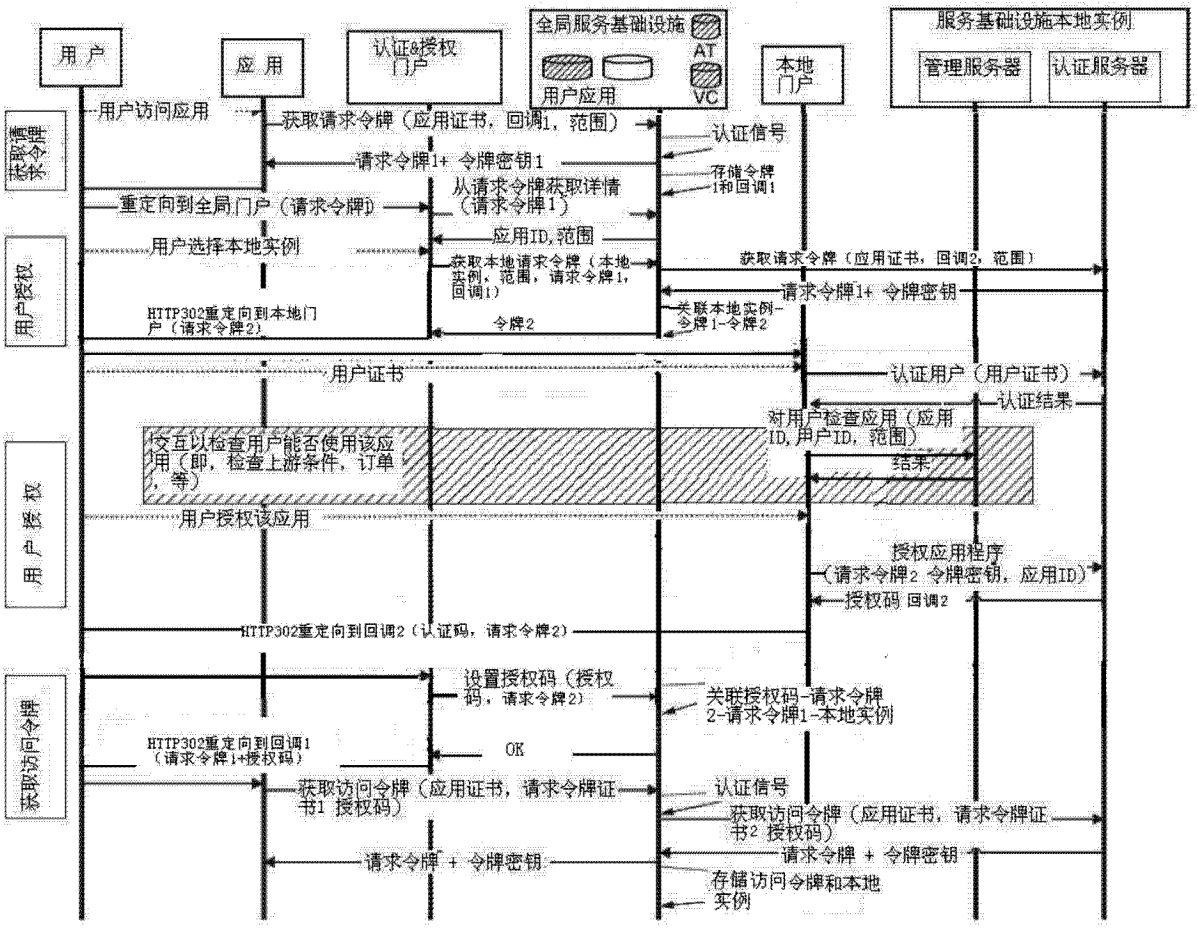

用户
应用
认证&授权门户
全局服务基础设施
AT
VC
用户应用
本地门户
服务基础设施本地实例
管理服务器
认证服务器
获取请求令牌
用户授权
用户授权
获取访问令牌
用户访问应用
获取请求令牌（应用证书，回调1，范围）
认证信号
请求令牌1+ 令牌密钥1
存储令牌1和回调1
从请求令牌获取详情（请求令牌1）
重定向到全局门户（请求令牌1）
应用ID,范围
用户选择本地实例
获取本地请求令牌（本地实例，范围，请求令牌1，回调1）
获取请求令牌（应用证书，回调2，范围）
请求令牌1+ 令牌密钥
关联本地实例-令牌1-令牌2
HTTP302重定向到本地门户（请求令牌2）
令牌2
用户证书
认证用户（用户证书）
认证结果
对用户检查应用（应用ID,用户ID，范围）
交互以检查用户能否使用该应用（即，检查上游条件，订单，等）
结果
用户授权该应用
授权应用程序（请求令牌2 令牌密钥，应用ID）
授权码 回调2
HTTP302重定向到回调2（认证码，请求令牌2）
设置授权码（授权码，请求令牌2）
关联授权码-请求令牌2-请求令牌1-本地实例
HTTP302重定向到回调1（请求令牌1+授权码）
OK
获取访问令牌（应用证书，请求令牌证书1 授权码）
认证信号
获取访问令牌（应用证书，请求令牌证书2 授权码）
请求令牌 + 令牌密钥
请求令牌 + 令牌密钥
存储访问令牌和本地实例

图 10

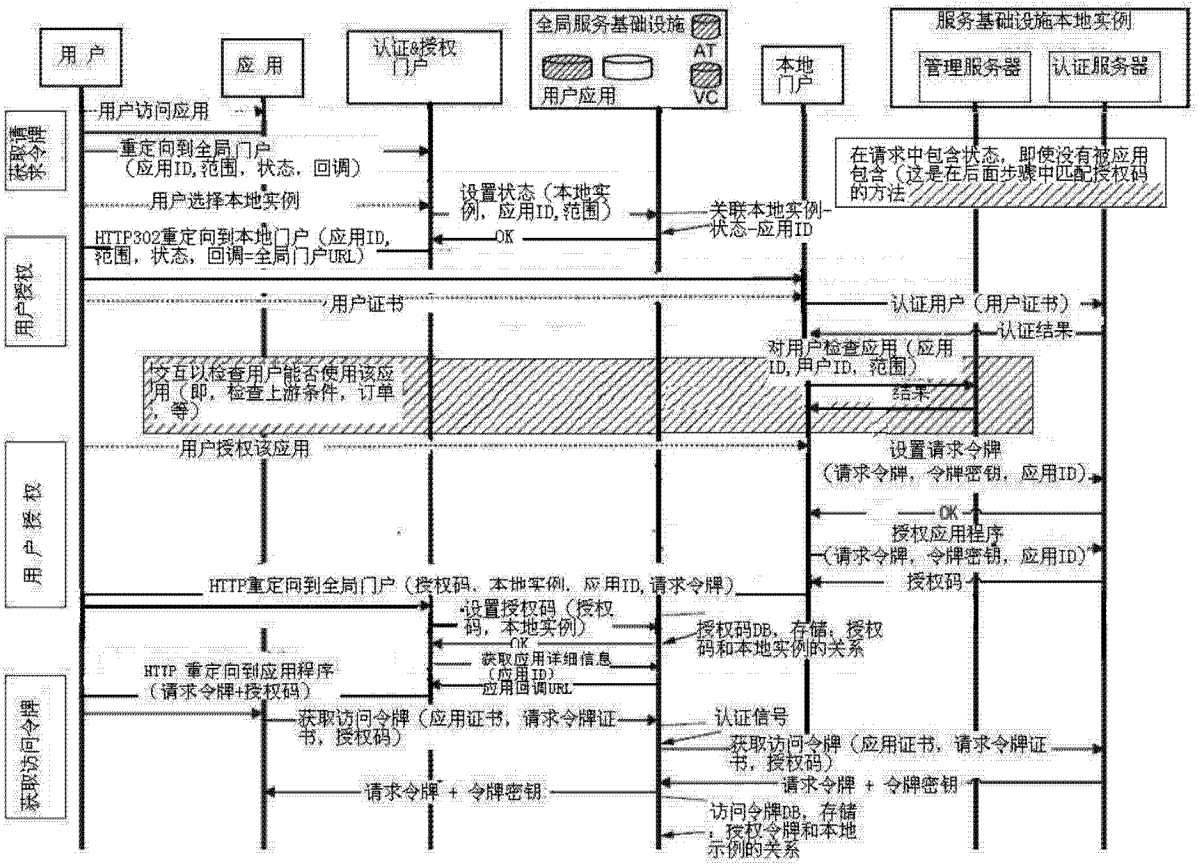

用户
应用
认证&授权门户
全局服务基础设施
AT
VC
用户应用
本地门户
服务基础设施本地实例
管理服务器
认证服务器
获取令牌
用户访问应用
重定向到全局门户（应用ID,范围，状态，回调）
用户选择本地实例
设置状态（本地实例，应用ID,范围）
关联本地实例-状态-应用ID
OK
HTTP302重定向到本地门户（应用ID,范围，状态，回调=全局门户URL）
在请求中包含状态，即使没有被应用包含（这是在后面步骤中匹配授权码的方法
用户授权
用户证书
认证用户（用户证书）
认证结果
对用户检查应用（应用ID,用户ID，范围）
交互以检查用户能否使用该应用（即，检查上游条件，订单，等）
结果
用户授权该应用
设置请求令牌（请求令牌，令牌密钥，应用ID）
OK
授权应用程序（请求令牌，令牌密钥，应用ID）
授权码
HTTP重定向到全局门户（授权码，本地实例，应用ID,请求令牌）
设置授权码（授权码，本地实例）
OK
授权码DB，存储，授权码和本地实例的关系
获取应用详细信息（应用ID）
应用回调URL
HTTP 重定向到应用程序（请求令牌+授权码）
获取访问令牌
获取访问令牌（应用证书，请求令牌证书，授权码）
认证信号
获取访问令牌（应用证书，请求令牌证书，授权码）
请求令牌 + 令牌密钥
请求令牌 + 令牌密钥
访问令牌DB，存储，授权令牌和本地示例的关系

图 11

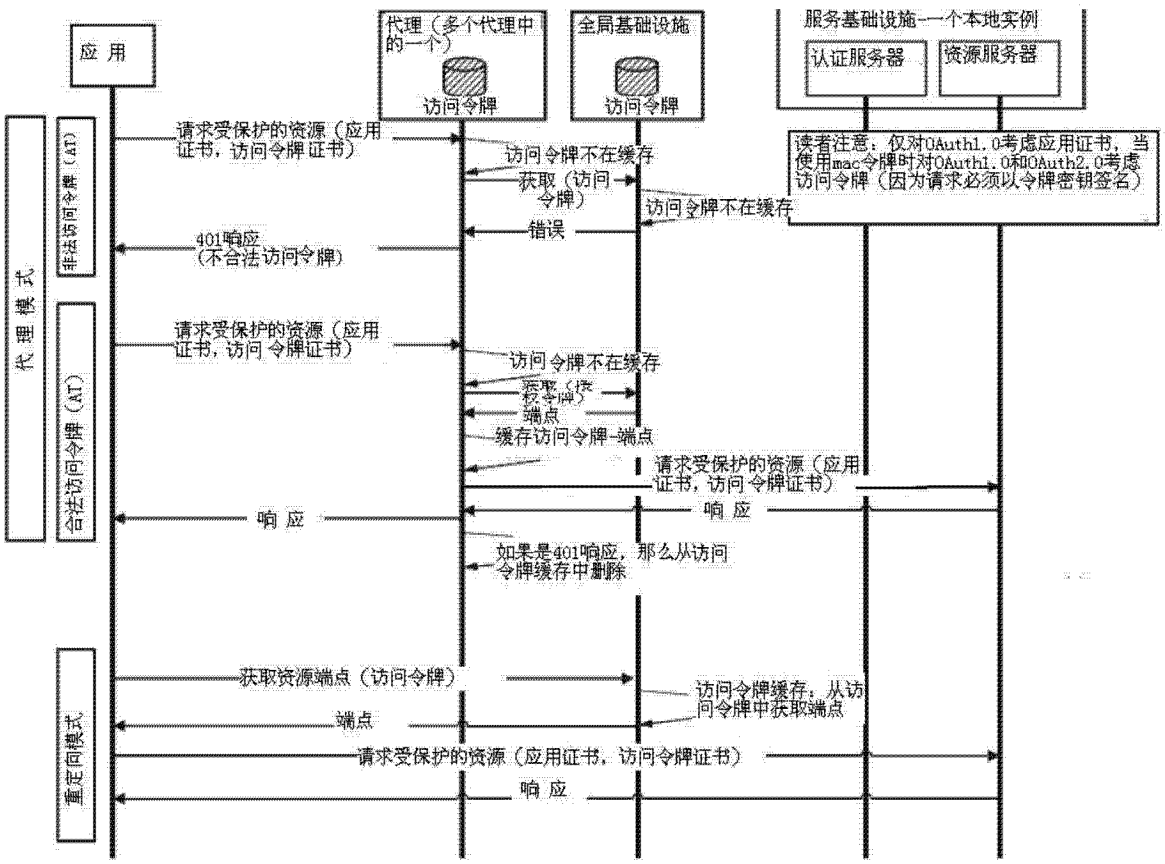

应 用
代理（多个代理中的一个）
访问令牌
全局基础设施
访问令牌
服务基础设施--一个本地实例
认证服务器
资源服务器
代理模式
非法访问令牌（AT）
合法访问令牌（AT）
重定向模式
请求受保护的资源（应用证书，访问令牌证书）
访问令牌不在缓存
获取（访问令牌）
访问令牌不在缓存
错误
401响应（不合法访问令牌）
读者注意：仅对OAuth1.0考虑应用证书，当使用mac令牌时对OAuth1.0和OAuth2.0考虑访问令牌（因为请求必须以令牌密钥签名）
请求受保护的资源（应用证书，访问令牌证书）
访问令牌不在缓存
获取（访问令牌）
端点
缓存访问令牌-端点
请求受保护的资源（应用证书，访问令牌证书）
响 应
响 应
如果是401响应，那么从访问令牌缓存中删除
获取资源端点（访问令牌）
访问令牌缓存，从访问令牌中获取端点
端点
请求受保护的资源（应用证书，访问令牌证书）
响 应

图 12

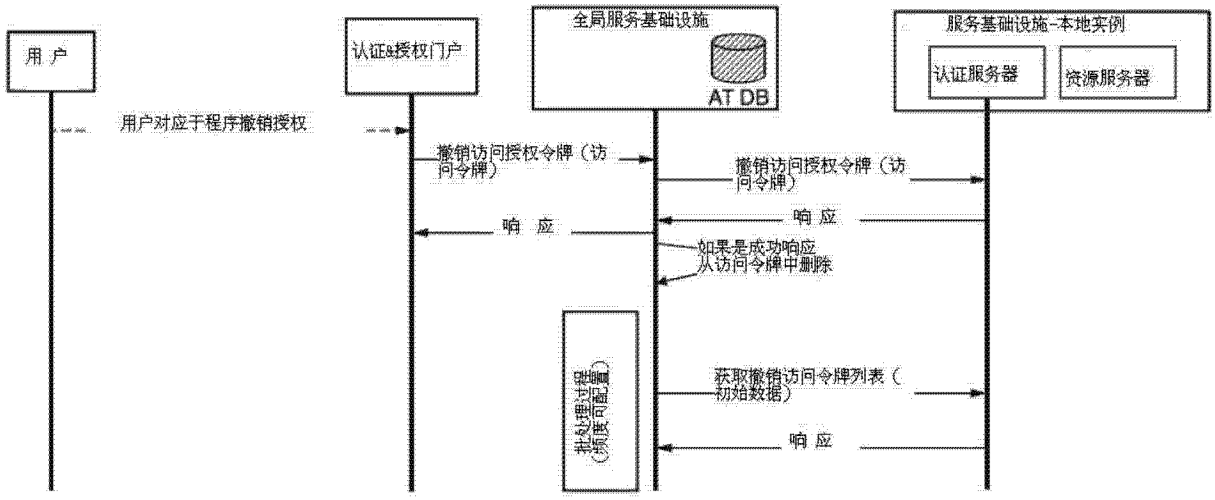

用 户
认证&授权门户
全局服务基础设施
AT DB
服务基础设施-本地实例
认证服务器
资源服务器
用户对应于程序撤销授权
撤销访问授权令牌（访问令牌）
撤销访问授权令牌（访问令牌）
响 应
响 应
如果是成功响应从访问令牌中删除
获取撤销访问令牌列表（初始数据）
响 应

图 13

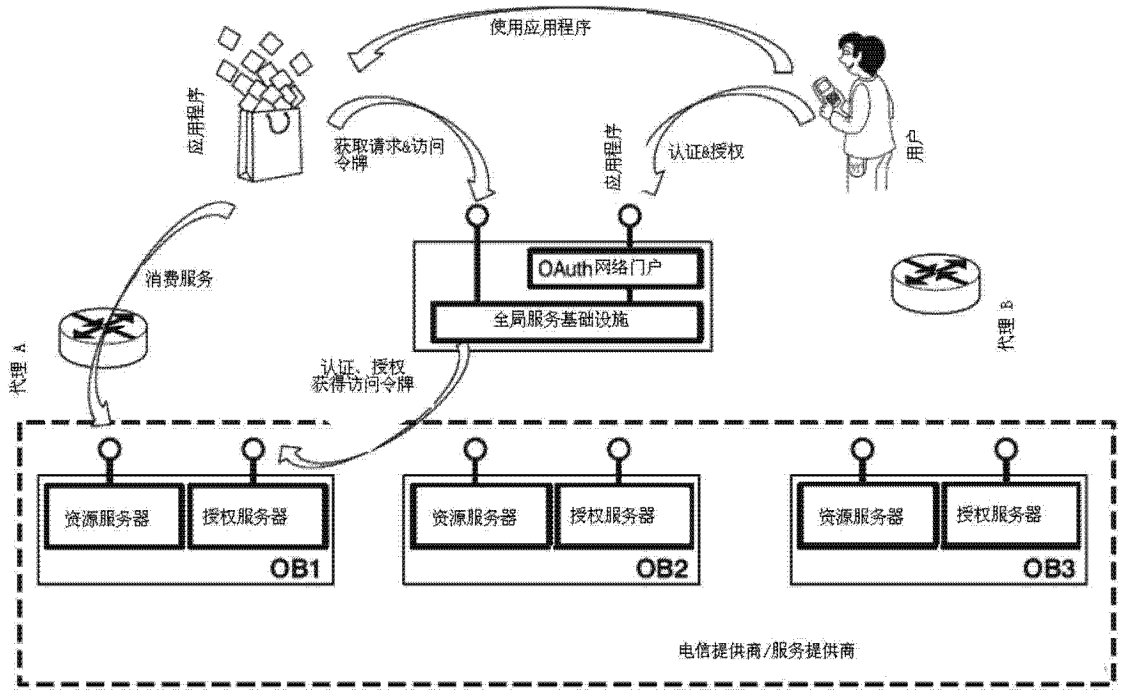

使用应用程序
应用程序
获取请求&访问令牌
应用程序
认证&授权
用户
消费服务
OAuth网络门户
全局服务基础设施
代理 A
代理 B
认证、授权
获得访问令牌
资源服务器
授权服务器
OB1
资源服务器
授权服务器
OB2
资源服务器
授权服务器
OB3
电信提供商/服务提供商

图 14