附录 B OAuth2 授权类型

本章内容

- OAuth2 密码授权
- OAuth2 客户凭证授权
- OAuth2 授权码授权
- OAuth2 隐式凭证授权
- OAuth2 令牌刷新

阅读了第 7 章，你可能会认为 OAuth2 看起来不太复杂。毕竟，你有一个认证服务，用于检查用户的凭证并将令牌发回给用户。每次用户想要调用由 OAuth2 服务器保护的服务时，都可以出示令牌。

不幸的是，现实世界并不简单。由于 Web 和基于云的应用程序具有相互关联的特性，用户期望他们能够安全地共享他们的数据，并在不同服务所拥有的不同应用程序之间集成功能。这从安全角度来看是一个独特的挑战，因为你希望跨不同的应用程序进行集成，而不是强迫用户与他们想要集成的每个应用程序共享凭证。

幸运的是，OAuth2 是一个灵活的授权框架，为应用程序提供了多种机制来验证和授权用户，而不用强制他们共享凭证。不幸的是，这也是 OAuth2 被认为是复杂的原因之一。 这些认证机制被称为认证授权。OAuth2 有四种形式的认证授权，客户端应用程序可以使用这些身份验证用户身份，接收访问令牌，然后验证该令牌。这些授权是：

- 密码
- 客户凭证
- 授权码

- 隐式

在接下来的部分中，我将介绍在执行每个 OAuth2 授权流程期间发生的活动。我也谈到何时使用一种授权类型。

B.1 密码授权

OAuth2 密码授权可能是要理解的最直接的授权类型。当应用程序和服务明确相互信任时使用此授权类型。例如，EagleEye Web 应用程序和 EagleEye Web 服务（许可和组织）都由 ThoughtMechanix 拥有，所以它们之间存在自然的信任关系。

> ***注意***：明确地说，当我提到“自然信任关系”时，我的意思是应用程序和服务完全由同一个组织拥有。它们在相同的策略和程序下管理。

当存在自然信任关系时，将 OAuth2 访问令牌暴露给调用应用程序几乎不用担心。例如，EagleEye Web 应用程序可以使用 OAuth2 密码授权来捕获用户凭证，并直接对 EagleEye OAuth2 服务进行身份验证。图 B.1 显示了 EagleEye 和下游服务之间的密码授权。

① Application owner registers application name with OAuth2 service, which provides a secret key

应用程序所有者将应用程序名称注册到提供密钥的 OAuth2 服务

② User logs into EagleEye, which passes user credentials with application name and key to OAuth2 service

用户登录到 EagleEye，它将具有应用程序名称和密钥的用户凭证传递给 OAuth2 服务

③ OAuth2 authenticates user and application and provides access token

OAuth2 认证用户和应用程序并提供访问令牌

④ *EagleEye attaches access token to any service calls from user*

EagleEye 将接入令牌附加到来自用户的任何服务调用

⑤ *Protected services call OAuth2 to validate access token*

受保护的服务调用 OAuth2 来验证访问令牌

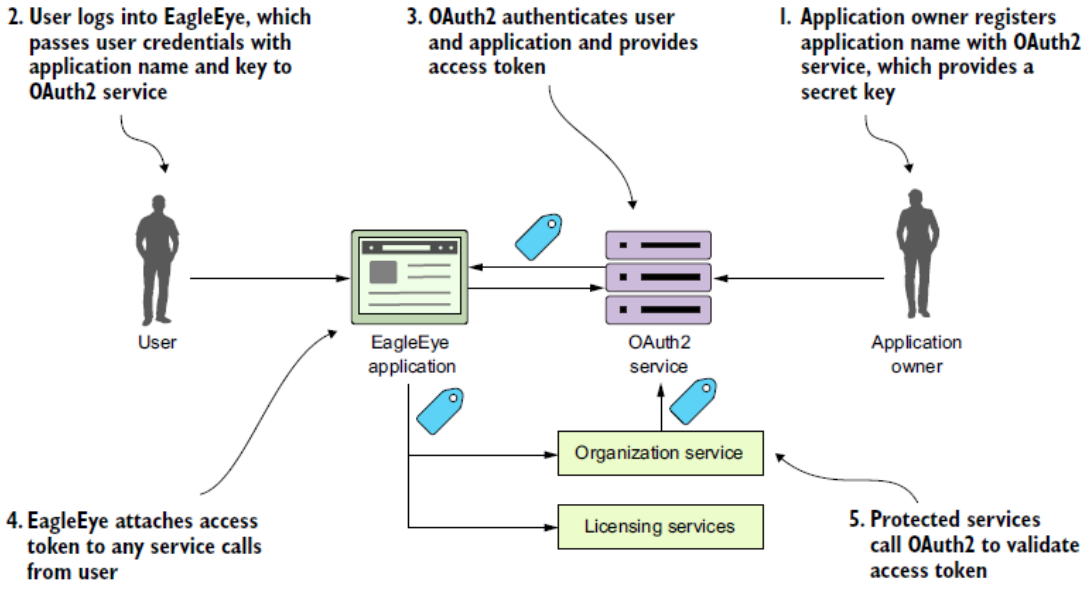

图 B.1 OAuth2 服务确定访问服务的用户是否是经过身份验证的用户。

在图 B.1 中，正在发生以下操作：

- 在 EagleEye 应用程序可以使用受保护资源之前，它需要在 OAuth2 服务中被唯一标识。通常，应用程序的所有者向 OAuth2 应用程序服务注册，并为其应用程序提供唯一的名称。然后 OAuth2 服务提供一个密钥返回到注册应用程序。

  应用程序的名称和 OAuth2 服务提供的密钥唯一标识尝试访问任何受保护资源的应用程序。

- 用户登录到 EagleEye 并将其登录凭证提供给 EagleEye 应用程序。EagleEye 将用户凭证以及应用程序名称/应用程序密钥直接传递给 EagleEye OAuth2 服务。

- EagleEye OAuth2 服务对应用程序和用户进行身份验证，然后将 OAuth2 访问令牌提供给用户。

- 每次 EagleEye 应用程序代表用户调用服务时，它都会传递 OAuth2 服务器提供的访问令牌。
- 当受保护的服务被调用（在这种情况下，许可服务和组织服务），服务回调 EagleEye OAuth2 服务来验证令牌。如果令牌是有效的，被调用的服务允许用户继续。如果令牌无效，则 OAuth2 服务返回 HTTP 状态码 403，表示令牌无效。

B.2 客户凭证授权

当应用程序需要访问受 OAuth2 保护的资源时，通常会使用客户凭证授权，但在交易中不涉及人。使用客户凭证授予类型，OAuth2 服务器仅基于应用程序名称和资源所有者提供的密钥进行身份验证。同样，当两个应用程序都属于同一个公司时，通常使用客户凭证工作。密码授权和客户凭证授权之间的区别在于客户凭证授权仅通过使用注册的应用程序名称和密钥进行认证。

例如，假设一小时一次，EagleEye 应用程序就会运行一个数据分析作业。作为其工作的一部分，它向 EagleEye 服务发出调用。但是，EagleEye 开发人员仍然希望应用程序在访问这些服务中的数据之前进行身份验证和授权。这是可以使用客户凭证授权的地方。图 B.2 显示了这个流程。

① Application owner registers data analytics job with OAuth2

应用程序所有者将数据分析作业注册到 OAuth2

② When the data analytics job runs, EagleEye passes application name and key to OAuth2

数据分析作业运行时，EagleEye 将应用程序名称和密钥传递给 OAuth2

③ OAuth2 authenticates application and provides access token

OAuth2 认证应用程序并提供访问令牌

④ EagleEye attaches access token to any service calls

EagleEye 将访问令牌附加到任何服务调用

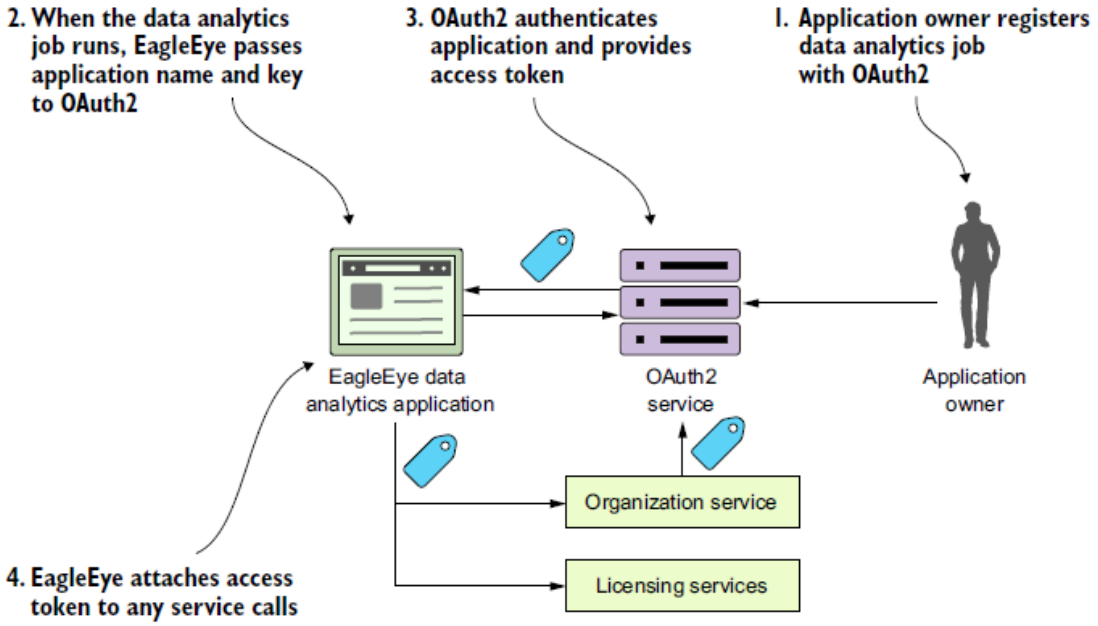

图 B.2 客户凭证授权是针对“无用户参与”的应用程序认证和授权。

- 资源所有者使用 OAuth2 服务注册 EagleEye 数据分析应用程序。资源所有者将提供应用程序名称并接收一个密钥。
- 当 EagleEye 数据分析作业运行时，它将出示其资源所有者提供的应用程序名称和密钥。
- EagleEye OAuth2 服务将使用应用程序名称和提供的密钥对应用程序进行身份验证，然后返回 OAuth2 访问令牌。
- 每次应用程序调用一个 EagleEye 服务时，都会出示通过服务调用收到的 OAuth2 访问令牌。

B.3 授权码授权

授权代码授权是迄今为止最复杂的 OAuth2 授权，但它也是最常用的流程，因为它允许来自不同供应商的不同应用程序共享数据和服务 ,而无需在多个应用程序中公开用户凭证。它还通过不让调用应用程序立即获得 OAuth2 访问令牌来执行额外的检查层 ,而是一个 “预访问” 授权码。

理解授权码授权的简单方法就是通过一个例子。假设你有一个也使用 Salesforce.com 的 EagleEye 用户。EagleEye 客户的 IT 部门已经构建了一个需要 EagleEye 服务（组织服务）数据的 Salesforce 应用程序。让我们来看图 B.3，看看授权码授权流程是如何工作的，以允许 Salesforce 从 EagleEye 组织服务中访问数据，而 EagleEye 客户不必将他们的 EagleEye 凭证暴露给 Salesforce。

① EagleEye user registers Salesforce application with OAuth2, obtains secret key and a callback URL to return users from EagleEye login to Salesforce.com.

EagleEye 用户使用 OAuth2 注册 Salesforce 应用程序，获取密钥和回调 URL，以将用户从 EagleEye 登录返回到 Salesforce.com。

② User configures Salesforce app with name, secret key, and a URL for the EagleEye OAuth2 login page.

用户使用 EagleEye OAuth2 登录页面的名称、密钥和 URL 配置 Salesforce 应用程序。

③ Potential Salesforce app users now directed to EagleEye login page; authenticated users return to Salesforce.com through callback URL (with authorization code).

潜在 Salesforce 应用程序用户现在指向 EagleEye 登录页面；经过身份验证的用户通过回调 URL（使用授权码）返回到 Salesforce.com。

④ Salesforce app passes authorization code along with secret key to OAuth2 and obtains access token.

Salesforce 应用程序将授权码和密钥一起传递给 OAuth2 并获取访问令牌。

⑤ Salesforce app attaches access token to any service calls.

Salesforce 应用程序将访问令牌附加到任何服务调用。

⑥ Protected services call OAuth2 to validate access token.

受保护的服务调用 OAuth2 来验证访问令牌。

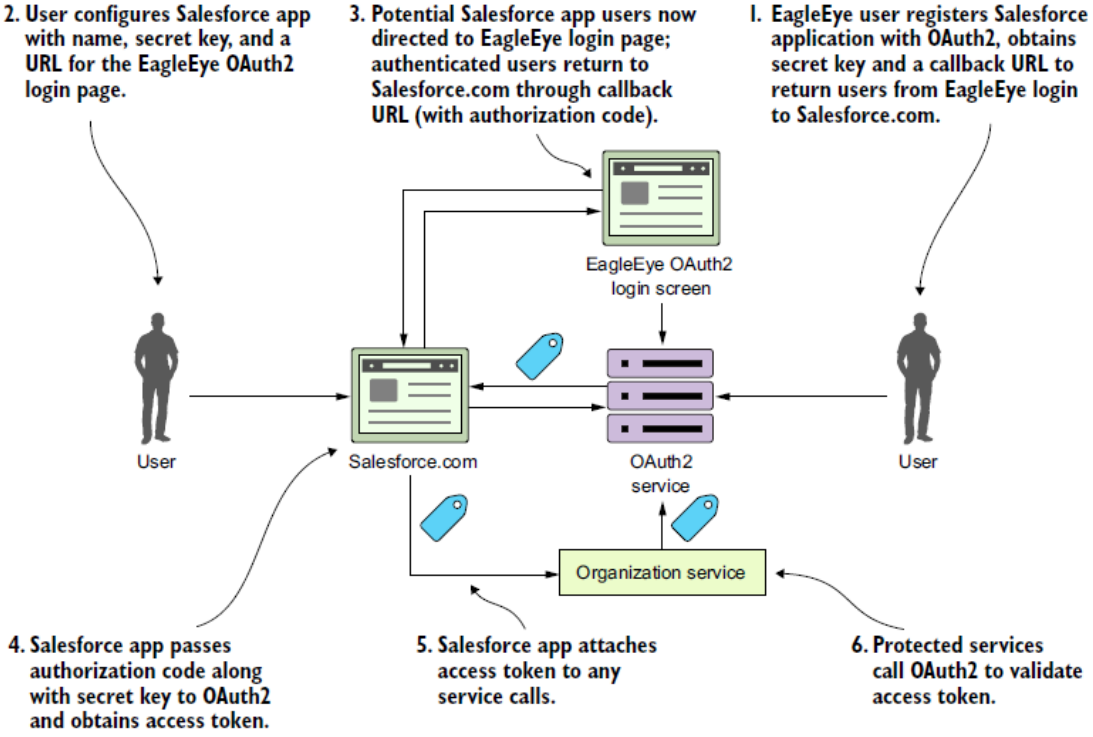

图 B.3 授权码授权允许应用程序共享数据而不公开用户凭证。

- EagleEye 用户登录到 EagleEye 并为其 Salesforce 应用程序生成应用程序名称和应用程序密钥。作为注册过程的一部分，它们还将会提供回调 Salesforce 应用程序的 URL。此回调 URL 是一个 Salesforce 网址，将在 EagleEye OAuth2 服务器验证用户的 EagleEye 凭证后被调用。
- 用户使用以下信息配置其 Salesforce 应用程序：

- ✓ 他们为 Salesforce 创建的应用程序名称
- ✓ 他们为 Salesforce 生成的密钥
- ✓ 指向 EagleEye OAuth2 登录页面的 URL
- ✓ 现在，当用户尝试使用他们的 Salesforce 应用程序并通过组织服务访问他们的 EagleEye 数据时，通过前面的项目符号描述的 URL，他们将被重定向到 EagleEye 登录页面。用户将提供他们的 EagleEye 凭证。如果他们提供了有效的 EagleEye 凭证，则 EagleEye OAuth2 服务器将生成一个授权码，并通过第 1 步中提供的 URL 将用户重定向到 SalesForce。授权码将作为回调 URL 上的查询参数发送。

- 自定义 Salesforce 应用程序将保留此授权码。注意：此授权码不是 OAuth2 访问令牌。
- 一旦授权码被存储，自定义的 Salesforce 应用程序就可以向 Salesforce 应用程序出示它们在注册过程中生成的密钥 ,并将授权码返回给 EagleEye OAuth2 服务器。EagleEye OAuth2 服务器将验证授权码是否有效，然后将 OAuth2 令牌返回给自定义 Salesforce 应用程序。每次自定义 Salesforce 需要对用户进行身份验证并获取 OAuth2 访问令牌时，都会使用此授权码。
- Salesforce 应用程序将调用 EagleEye 组织服务，在 header 头中传递 OAuth2 令牌。
- 组织服务将使用 EagleEye OAuth2 服务验证传递到 EagleEye 服务调用的 OAuth2 访问令牌。如果令牌有效，组织服务将处理用户的请求。

哇！我需要呼吸空气。应用程序到应用程序的集成是错综复杂的。从整个流程中要注意的是，即使用户登录到 Salesforce 并且正在访问 EagleEye 数据，用户的 EagleEye 凭证也

不会直接暴露给 Salesforce。在 EagleEye OAuth2 服务生成并提供初始授权代码之后，用户不必将其凭证提供回 EagleEye 服务。

B.4 隐式授权

授权码授权在通过传统的服务器端 Web 编程环境（如 Java 或.NET）运行 Web 应用程序时使用。如果你的客户端应用程序是纯粹的 JavaScript 应用程序或完全在 Web 浏览器中运行并且不依赖服务器端调用来调用第三方服务的移动应用程序，会发生什么情况？

这是最后一个授权类型，即隐式授权，开始发挥作用。图 B.4 显示了隐式授权中发生的一般流程。

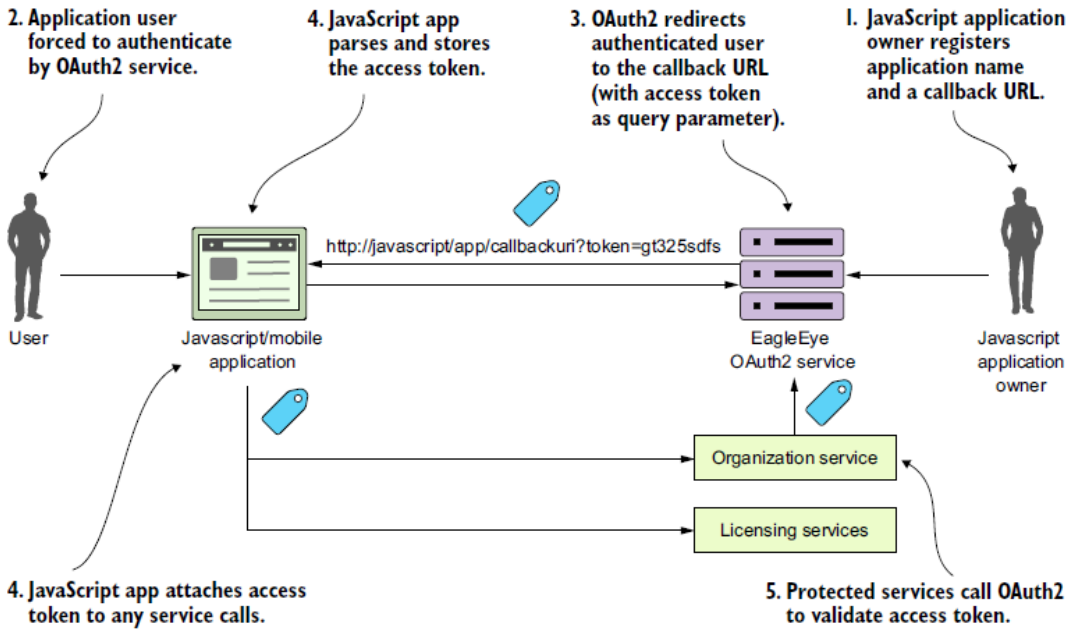

图 B.4 隐式授权用于基于浏览器的单页面应用程序（SPA）JavaScript 应用程序。

① *JavaScript application owner registers application name and a callback URL.*

JavaScript 应用程序所有者注册应用程序名称和回调 URL。

② *Application user forced to authenticate by OAuth2 service.*

应用程序用户被强制通过 OAuth2 服务进行身份验证。

③ OAuth2 redirects authenticated user to the callback URL (with access token as query parameter).

OAuth2 将经过身份验证的用户重定向到回调 URL（使用访问令牌作为查询参数）。

④ JavaScript app parses and stores the access token.

JavaScript 应用程序解析并存储访问令牌。

⑤ JavaScript app attaches access token to any service calls.

JavaScript 应用程序将访问令牌附加到任何服务调用。

⑥ Protected services call OAuth2 to validate access token.

受保护的服务调用 OAuth2 来验证访问令牌。

使用隐式授权，通常使用完全在浏览器内运行的纯 JavaScript 应用程序。在其它流程中，客户端正在与执行用户请求的应用程序服务器进行通信，而应用程序服务器正在与任何下游服务进行交互。使用隐式授权类型，所有服务交互都直接从用户的客户端（通常是 Web 浏览器）发生。在图 B.4 中，正在进行以下操作：

- JavaScript 应用程序的所有者已经使用 EagleEye OAuth2 服务器注册了应用程序。他们提供了一个应用程序名称以及一个回调 URL，该 URL 将与用户的 OAuth2 访问令牌一起被重定向。
- JavaScript 应用程序将调用 OAuth2 服务。JavaScript 应用程序必须提供预先注册的应用程序名称。OAuth2 服务器将强制用户进行身份验证。
- 如果用户成功进行身份验证，则 EagleEye OAuth2 服务将不会返回令牌，而是将用户重定向回到在第一步中注册的 JavaScript 应用程序所有者的页面。在重定向到的 URL 中，OAuth2 访问令牌将作为查询参数通过 OAuth2 认证服务传递。
- 应用程序将接收传入的请求并运行 JavaScript 脚本，该脚本将解析 OAuth2 访问

令牌并将其存储（通常为 Cookie）。

- 每次调用受保护的资源时，OAuth2 访问令牌都会出示给被调用的服务。
- 被调用的服务将验证 OAuth2 令牌，并检查用户是否有权执行他们正在尝试执行的操作。

关于 OAuth2 隐式授权请记住几点：

- 隐式授权是 OAuth2 访问令牌直接暴露给公共客户端（Web 浏览器）的唯一授予类型。在授权码授权中，客户端应用程序获取返回到托管应用程序的应用程序服务器的授权码。通过授权码授权，用户可以通过出示授权码来授予 OAuth2 访问权限。返回的 OAuth2 令牌不会直接暴露给用户的浏览器。

  在客户凭证授权中，授权发生在两个基于服务器的应用程序之间。在密码授权中，请求服务的应用程序和服务都是可信的，并且由同一个组织拥有。
- 由隐式授权生成的 OAuth2 令牌更容易受到攻击和滥用，因为令牌可供浏览器使用。在浏览器中运行的任何恶意 JavaScript 都可以访问 OAuth2 访问令牌，并以你的名义调用你检索到 OAuth2 令牌的服务，并基本上是模仿你。
- 隐式授权类型 OAuth2 令牌应该是短暂的（1-2 小时）。因为 OAuth2 访问令牌存储在浏览器中，所以 OAuth2 规范（和 Spring Cloud 安全）不支持可以自动更新令牌的刷新令牌的概念。

B.5 如何刷新令牌

当 OAuth2 访问令牌被发布时，它的有效时间有限，并且最终将过期。当令牌到期时，调用应用程序（和用户）将需要使用 OAuth2 服务重新进行身份验证。但是，在大多数 Oauth2 授权流程中，OAuth2 服务器将同时发布一个访问令牌和一个刷新令牌。客户端可

以将刷新令牌提供给 OAuth2 认证服务，服务将验证刷新令牌，然后发布新的 OAuth2 访问令牌。我们来看图 B.5，并浏览刷新令牌流程：

- 用户已经登录到 EagleEye，并且已经使用 EagleEye OAuth2 服务进行了身份验证。用户很高兴地工作，但不幸的是他们的令牌已经过期。
- 用户下一次尝试调用服务（比如组织服务）时，EagleEye 应用程序会将已过期的令牌传递给组织服务。
- 组织服务将尝试使用 OAuth2 服务验证令牌，该服务将返回 HTTP 状态代码 401（未授权）和 JSON 有效载荷，表明该令牌不再有效。组织服务会将 HTTP 401 状态码返回给调用服务。
- EagleEye 应用程序获取 401 HTTP 状态码和表明从组织服务返回调用失败原因的 JSON 有效载荷。EagleEye 应用程序将使用刷新令牌调用 OAuth2 认证服务。OAuth2 认证服务将验证刷新令牌，然后发回新的访问令牌。

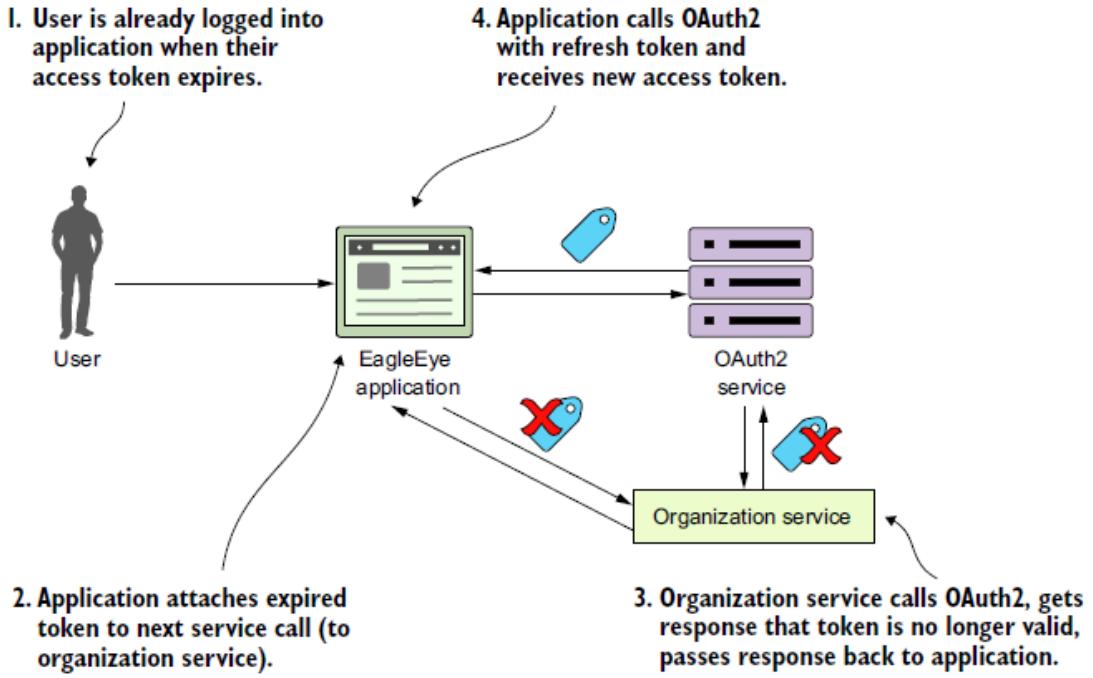

图 B.5 刷新令牌流程允许应用程序获取新的访问令牌，而不会强制用户重新进行身份验证。

① User is already logged into application when their access token expires.

当访问令牌到期时，用户已经登录到应用程序。

② *Application attaches expired token to next service call (to organization service).*

应用程序将过期令牌附加到下一个服务调用（到组织服务）。

③ *Organization service calls OAuth2, gets response that token is no longer valid, passes response back to application.*

组织服务调用 OAuth2，获取该令牌不再有效的响应，并将响应传递回应用程序。

④ *Application calls OAuth2 with refresh token and receives new access token.*

应用程序使用刷新令牌调用 OAuth2 并接收新的访问令牌。