授权 OAuth 应用程序

您可以让其他用户授权您的 OAuth 应用程序。

本文内容

Web 应用程序流程
设备流程
非 Web 应用程序流程
重定向 URL
为 OAuth 应用程序创建多个令牌
指示用户审查其访问权限
疑难解答

GitHub 的 OAuth 实现支持标准授权代码授予类型以及 OAuth 2.0 设备授权授予（针对无法访问 web 浏览器的应用程序）。

如果您想要跳过以标准方式授权应用程序，例如测试应用程序时， 您可以使用非 web 应用程序流程。

要授权您的 OAuth 应用程序，请考虑哪个授权流程最适合您的应用程序。

- Web 应用程序流程：用于授权在浏览器中运行标准 OAuth 应用程序的用户。 （不支持隐式授予类型。）
- 设备流程：用于无头应用程序，例如 CLI 工具。

Web 应用程序流程

注：如果您要构建 GitHub 应用程序，依然可以使用 OAuth web 应用程序流程，但设置方面有一些重要差异。更多信息请参阅“识别和授权 GitHub 应用程序用户”。

授权应用程序用户的 web 应用程序流程是：

1. 用户被重定向，以请求他们的 GitHub 身份

2 用户被 GitHub 重定向回您的站点

3 您的应用程序使用用户的访问令牌访问 API

1. 请求用户的 GitHub 身份

```
GET https://github.com/login/oauth/authorize
```

当您的 GitHub 应用程序指定 `login` 参数后，它将提示拥有特定账户的用户可以用来登录和授权您的应用程序。

参数

名称	类型	描述
`client_id`	`字符串`	**必填**。 您注册 时从 GitHub 收到的客户端 ID。
`redirect_uri`	`字符串`	用户获得授权后被发送到的应用程序中的 URL。 有关重定向 url，请参阅下方的详细信息。
`login`	`字符串`	提供用于登录和授权应用程序的特定账户。
`作用域`	`字符串`	用空格分隔的作用域列表。 如果未提供，对于未向应用程序授权任何作用域的用户，`scope` 将默认为空白列表。 对于已向应用程序授权作用域的用户，不会显示含作用域列表的 OAuth 授权页面。 相反，通过用户向应用程序授权的作用域集，此流程步骤将自动完成。 例如，如果用户已执行两次 web 流程，且授权了一个拥有 `user` 作用域的令牌和一个拥有 `repo` 作用域的令牌，未提供 `scope` 的第三次 web 流程将收到拥有 `user` 和 `repo` 作用域的令牌。
`state`	`字符串`	不可猜测的随机字符串。 它用于防止跨站请求伪造攻击。

2. 用户被 GitHub 重定向回您的站点

如果用户接受您的请求，GitHub 将重定向回您的站点，其中，代码参数为临时 `code`，`state` 参数为您在上一步提供的状态。 临时代码将在 10 分钟后到期。 如果状态不匹配，然后第三方创建了请求，您应该中止此过程。

用此 `code` 换访问令牌：

```
POST https://github.com/login/oauth/access_token
```

参数

名称	类型	描述
`client_id`	字符串	**必填。**您从 GitHub 收到的 OAuth 应用程序 的客户端 ID。
`client_secret`	字符串	**必填。**您从 GitHub 收到的 OAuth 应用程序 的客户端密钥。
`代码`	字符串	**必填。**您收到的响应第 1 步的代码。
`redirect_uri`	字符串	用户获得授权后被发送到的应用程序中的 URL。

响应

默认情况下，响应采用以下形式：

```
access_token=e72e16c7e42f292c6912e7710c838347ae178b4a&token_type=bearer
```

您也可以根据“接受”标头接收不同格式的内容：

```
Accept: application/json
{"access_token":"e72e16c7e42f292c6912e7710c838347ae178b4a", "scope":"repo,gist", "token

Accept: application/xml
<OAuth>
  <token_type>bearer</token_type>
  <scope>repo,gist</scope>
  <access_token>e72e16c7e42f292c6912e7710c838347ae178b4a</access_token>
</OAuth>
```

3. 使用访问令牌访问 API

访问令牌可用于代表用户向 API 提出请求。

```
Authorization: token OAUTH-TOKEN
GET https://api.github.com/user
```

例如，您可以像以下这样在 curl 中设置“授权”标头：

```
curl -H "Authorization: token OAUTH-TOKEN" https://api.github.com/user
```

设备流程

设备流程允许您授权用户使用无头应用程序，例如 CLI 工具或 Git 凭据管理器。

设备流程概述

1. 您的应用程序会请求设备和用户验证码，并获取用户将在其中输入用户验证码的授权 URL。
2. 应用程序提示用户在 `https://github.com/login/device` 中输入用户验证码。
3. 应用程序轮询用户身份验证状态。 用户授权设备后，应用程序将能够使用新的访问令牌进行 API 调用。

第 1 步：应用程序从 GitHub 请求设备和用户验证码

```
POST https://github.com/login/device/code
```

您的应用程序必须请求用户验证码和验证 URL，因为应用程序在下一步中提示用户进行身份验证时将使用它们。 此请求还返回设备验证代码，应用程序必须使用它们来接收访问令牌和检查用户身份验证的状态。

输入参数

名称	类型	描述
`client_id`	字符串	**必填。**您从 GitHub 收到的应用程序客户端 ID。
作用域	字符串	应用程序请求访问的范围。

响应

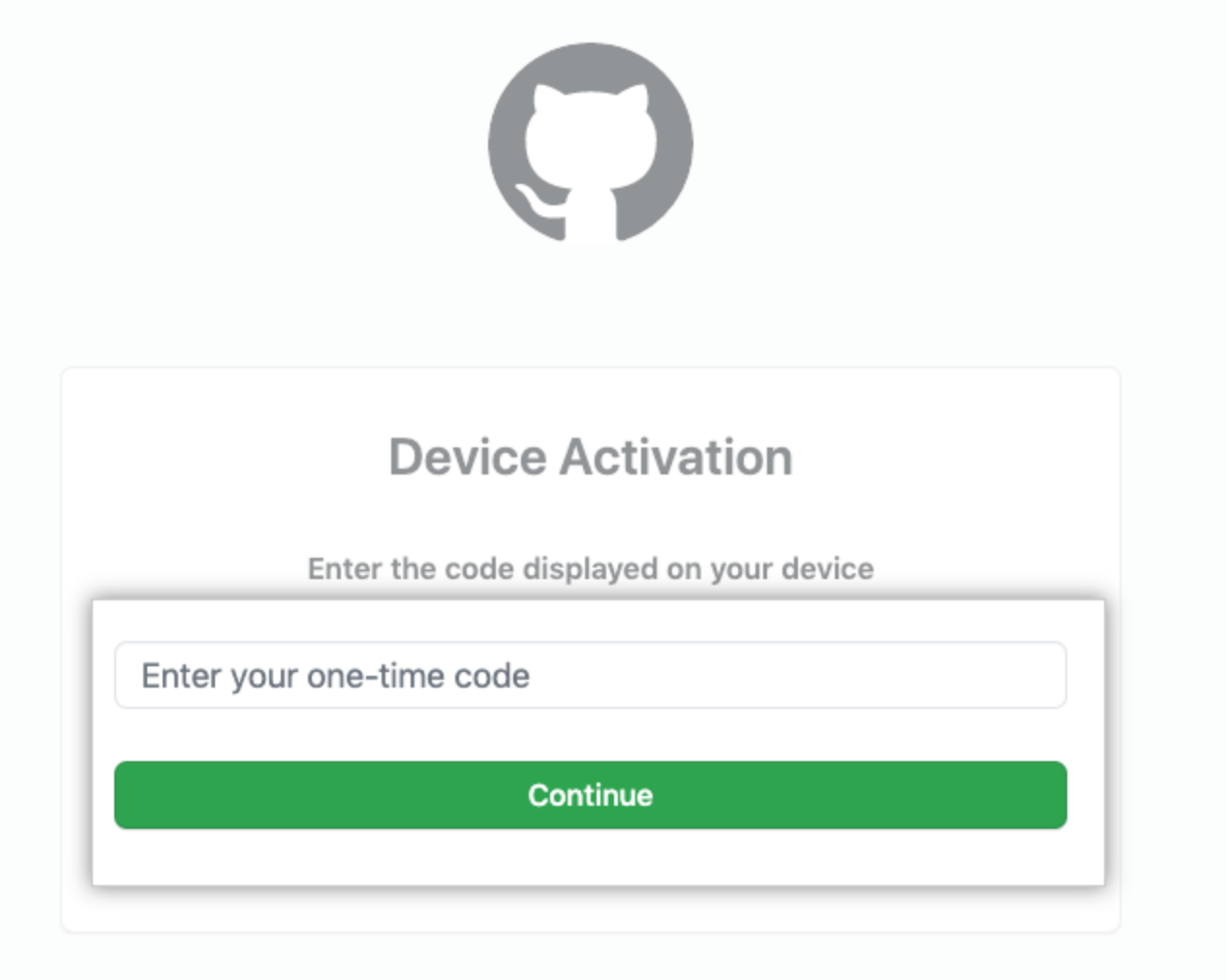
```
{
  "device_code": "3584d83530557fdd1f46af8289938c8ef79f9dc5",
  "user_code": "WDJB-MJHT",
  "verification_uri": "https://github.com/login/device",
  "expires_in": 900,
  "interval": 5
}
```

响应参数

名称	类型	描述
`device_code`	`字符串`	设备验证码为 40 个字符，用于验证设备。
`user_code`	`字符串`	用户验证码显示在设备上，以便用户可以在浏览器中输入该代码。 此代码为 8 个字符，中间有连字符。
`verification_uri`	`字符串`	用户需要在其中输入 `user_code` 的验证 URL：`https://github.com/login/device`。
`expires_in`	`整数`	`device_code` 和 `user_code` 到期前的秒数。 默认值为 900 秒或 15 分钟。
`间隔`	`整数`	发出新的访问令牌请求 (`POST https://github.com/login/oauth/access_token`) 以完成设备授权之前必须经过的最小秒数。 例如，如果间隔为 5，则只有经过 5 秒后才能发出新请求。 如果您在 5 秒内发出多个请求，则将达到速率限制并收到 `slow_down` 错误。

第 2 步：提示用户在浏览器中输入用户代码

您的设备将显示用户验证码并提示用户在 `https://github.com/login/device` 中输入该代码。



第 3 步：应用程序轮询 GitHub 以检查用户是否授权设备

```
POST https://github.com/login/oauth/access_token
```

应用程序将发出设备授权请求以轮询 `POST https://github.com/login/oauth/access_token`，直到设备和用户代码到期或者用户已使用有效的用户代码成功授权该应用程序。 应用程序必须使用在第 1 步中检索到的最小轮询 `interval`，以避免速率限制错误。 更多信息请参阅“设备流程的速率限制”。

用户必须在 15 分钟（或 900 秒内）内输入有效代码。 15 分钟后，您需要使用 `POST https://github.com/login/device/code` 请求新的设备授权代码。

一旦用户授权， 应用程序将收到一个访问令牌，该令牌可用于代表用户向 API 发出请求。

输入参数

名称	类型	描述
`client_id`	字符串	**必填。**您从 GitHub 收到的 OAuth 应用程序 的客户端 ID。
`device_code`	字符串	**必填。**您从 `POST https://github.com/login/device/code` 请求收到的设备验证码。

名称	类型	描述
`grant_type`	字符串	**必填。**授予类型必须是 `urn:ietf:params:oauth:grant-type:device_code`。

响应

```
{
 "access_token": "e72e16c7e42f292c6912e7710c838347ae178b4a",
  "token_type": "bearer",
  "scope": "user"
}
```

设备流程的速率限制

当用户在浏览器上提交验证码时，每个应用程序在一个小时内的提交速率限制为 50 个。

如果您在请求之间所需的最短时间段（或 `interval`）内发出多个访问令牌请求（`POST https://github.com/login/oauth/access_token`），您将达到速率限制并收到 `slow_down` 错误响应。 `slow_down` 错误响应将给最近的 `间隔` 增加 5 秒。 更多信息请参阅“设备流程的错误”。

设备流程的错误代码

错误代码	描述
`authorization_pending`	授权请求待处理并且用户尚未输入用户代码时，将发生此错误。 应用程序应该继续轮询 `POST https://github.com/login/oauth/access_token`，但不会超过 `interval`，它规定了每个请求之间的最小秒数。
`slow_down`	当您收到 `slow_down` 错误时，使用 `POST https://github.com/login/oauth/access_token` 请求之间的所需最小 `interval` 或时间段将额外增加 5 秒。 例如，如果两次请求之间的开始间隔至少需要 5 秒，并且您收到了 `slow_down` 错误响应，则现在必须等待至少 10 秒才能发出新的 OAuth 访问令牌请求。 错误响应包括您必须使用的新 `interval`。
`expired_token`	如果设备代码已过期，您将会看到 `token_expendent` 错误。 您必须发出新的设备代码请求。
`unsupported_grant_type`	授予类型必须为 `urn:ietf:params:oauth:grant-type:device_code`，并在您轮询 OAuth 令牌请求 `POST https://github.com/login/oauth/access_token` 时作为输入参数包括在内。
`incorrect_client_credentials`	对于设备流程，您必须传递应用程序的客户端 ID，您可以在应用程序设置页面上找到该 ID。 设备流程不需要 `client_secret`。

错误代码	描述
`incorrect_device_code`	提供的 device_code 无效。
`access_denied`	当用户在授权过程中单击取消时，您将收到 `access_denied` 错误，用户将无法再次使用验证码。

更多信息请参阅“OAuth 2.0 设备授权授予”。

非 Web 应用程序流程

非 web 身份验证适用于测试等有限的情况。 如果您需要，可以使用基本验证，通过个人访问令牌设置页面创建个人访问令牌。 此方法支持用户随时撤销访问权限。

> **注：**使用非 web 应用流程创建 OAuth2 令牌时，如果您或您的用户已启用双重身份验证，请确保明白如何使用双重身份验证。

重定向 URL

`redirect_uri` 参数是可选参数。 如果遗漏，GitHub 则将用户重定向到 OAuth 应用程序设置中配置的回调 URL。 如果提供，重定向 URL 的主机和端口必须完全匹配回调 URL。 重定向 URL 的路径必须引用回调 URL 的子目录。

```
CALLBACK: http://example.com/path

GOOD: http://example.com/path
GOOD: http://example.com/path/subdir/other
BAD:  http://example.com/bar
BAD:  http://example.com/
BAD:  http://example.com:8080/path
BAD:  http://oauth.example.com:8080/path
BAD:  http://example.org
```

本地主机重定向 URL

可选的 `redirect_uri` 参数也可用于本地主机 URL。 如果应用程序指定 URL 和端口，授权后，应用程序用户将被重定向到提供的 URL 和端口。 `redirect_uri` 不需要匹配应用程序回调 url 中指定的端口。

对于 `http://localhost/path` 回调 URL，您可以使用此 `redirect_uri`：

```
http://localhost:1234/path
```

为 OAuth 应用程序创建多个令牌

您可以为用户/应用程序/作用域组合创建多个令牌，以便为特定用例创建令牌。

如果您的 OAuth 应用程序支持一个使用 GitHub 登录且只需基本用户信息的工作流程，此方法将非常有用。 另一个工作流程可能需要访问用户的私有仓库。 您的 OAuth 应用程序可以使用多个令牌为每个用例执行 web 流程，只需要所需的作用域。 如果用户仅使用您的应用程序登录，则无需向他们的私有仓库授予您的 OAuth 应用程序访问权限。

每个用户/应用程序/作用域组合签发的令牌数量有限。 如果您的应用程序请求足够的令牌超越其中一个限制，*所请求的作用域相同的*旧令牌将停止工作。

警告: 从 OAuth 应用程序 撤销所有权限将会删除应用程序代表用户生成的 SSH 密钥，包括部署密钥。

指示用户审查其访问权限

您可以链接至 OAuth 应用程序的授权信息，以便用户审查和撤销其应用程序授权。

要构建此链接，需要使用注册应用程序时从 GitHub 收到的 OAuth 应用程序 `client_id`。

```
https://github.com/settings/connections/applications/:client_id
```

提示：要详细了解您的 OAuth 应用程序可以为用户访问的资源，请参阅“为用户发现资源。”

疑难解答

- "对授权请求错误进行故障排除"
- "对 OAuth 应用程序访问令牌请求错误进行故障排除"
- "设备流程错误"