# Các khai niệm

## Roles

OAuth định nghĩa 4 roles:

resource owner – Một entity có khả năng cấp truy cập tới tài nguyên được bảo vệ

resource server – server hosting các tài nguyên được bảo vệ

client – một application request tài nguyên được bảo vệ trên danh nghĩa của resource owner và có được sự cho phép của nó

authorization server – The sever ban phát access tokens tới client sau khi được resource owner xác minnh thành công.

## Các khai niệm nhỏ

### Authorization Grant (cấp ủy quyền) – là một chứng chỉ đại diện việc ủy quyền của chủ sở hữu tài nguyên và nó được sử dụng bởi client để lấy các tài nguyên trên danh nghĩa chủ tài nguyên

#### Authorization Code – đạt được bởi việc sử dụng authorization server như là một người trung gian giữa client và resource owner.Thay vì request việc ủy quyền một cách trực tiếp từ resource owner, the client điều hướng resource owner tới authorization server, sau đó nó lại điều hướng resource owner quay trở lại client với authorization code (code ở đây như là gift code, 1 đoạn mã nhập vào để nhận quà chứ k phải source code). Bởi vì resource owner xác thực trên authorization server nên các chứng chỉ của resource owner (các thông tin của resource owner nhập để xác thực) không bao giờ được share với client. the authorization code cung cấp vài lợi ích bảo mật quan trọng như là khả năng xác thực client (tức là xác thực luôn thằng application) cũng như là truyền access token trực tiếp tới client mà không cần thông qua resource owner’ user-agent (tức kiểu như là một entity cung cấp thông tin lquan đến trình duyệt, hệ điều hành...)và có khả năng phơi nó ra tới cái khác nhìn thấy, bao gồm cả user.

#### Implicit – The implicit grant là một flow authorization ccode giản thể được tới ưu cho clients thực thi trên browser bằng việc sử dụng script language. Trong flow này, thay vì phát cho client một authorization code thì client được phát cho một access token trực tiếp, và khi dùng cách này, authorization server không xác thực client (vấn đề bảo mật ở đây là không an toàn, vì chỉ cần ai lấy được token thì họ chỉ cần dùng client bất kì để sử dụng). Vì vậy trong các trường hợp này, danh tính client được xác minh thông qua redirection URI được sử dụng để chuyển access token tới client

## Resource owner

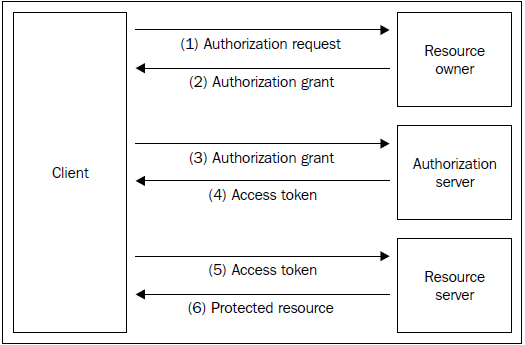
A resource owner là một entity mà có khả năng truy cập vào protected resource. Khi entity này là một người thì khai niệm end user và user có thể được sử dụng thay thế

## Authorization server

Để một client có thể truy cập vào protected resources, thì client phải được cấp quyền (authorized) bởi resource owner.

Vậy xin quyền ở đâu để được câp, thì ở đây the authorization server sẽ làm việc này, đóng vai trò làm trụ sở cấp – nó hỏi user (resource owner) để xác nhận client sẽ được cấp quyền để truy cập. Với mỗi authorization request thành công (cấp quyền thành công), server sẽ đưa cho client một thứ được gọi là access token (kiểu như là sau khi xin đi nước ngoài sẽ được cấp visa, visa ở đây tương ứng như access token). token đã được phát thì chỉ định tới client-application đã được cấp quyền, nên access token đại diện cho mối quan hệ giữa application server, client application và resource owner

## Resource server



Resource server là một người phục vụ protected resources, cái này được truy cập bởi các request đã được cấp quyền từ client application

Cách mà the authorization và resuource servers giao tiếp thì ngoài phạm vi của OAuth 2.0 vì vậy có vài implementations trong thực tế.

## Client

Một client là một application cái mà tạo các request tới protected resources trên resource server. Một client application có thể là một web app, desktop app,...

## Authorization flow

## Abstract example

Hay còn gọi là theo kiểu mô hình nguyên lý hoạt động chung.

Ví dụ như diagram bên trên là một abstract example

## OAuth 2.0 grant flows (tức là các flow cấp quyền)

Một trong các cải tiến đáng kể trong OAuth 2.0 so với OAuth 1.0 là việc định nghĩa của các grant flows. Mỗi flow được đề xuấy trong một vài kịch bản, chúng ta sẽ xem qua các flow được dùng trong bối cảnh nào.

* Authorization code grant flow – thằng này được sử dụng cho web server applications và nó là grant được sử dụng thường xuyên nhất
* Implicit grant flows – được sử dụng cho client applications cái mà không có khả năng giữ an toàn credentials và nó tường được sử dụng cho app chỉ cần đọc dữ liệu
* Resource owner password credentials grant flows – chỉ là một grant flow mà ở đó username và pass từ resource owner được gửi đi trong việc đổi một access toke, nó thường được sử dụng trong bối cảnh mà có một solution có sẵn đang migrating lên trên OAuth 2.0
* Client credentials grant flow – được sử dụng cho client app muốn request access tới một vài service provider, thay mặt cho chính bản thân chúng mà không phải cho một vài end user, nó sử dụng khi cần truy cập đến nhiều service APIs

## Tokens

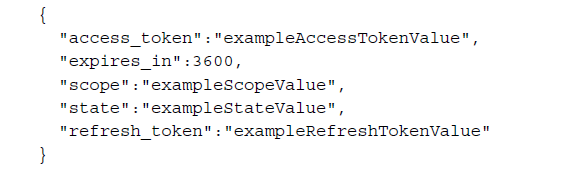
Có 2 kiểu trong OAuth 2.0 là access token và refresh token. the access token là một cái được sử dụng bởi client application khi tạo một requests tới protected resources, còn refresh token thì là một thằng được sử dụng để làm mới access token khi nó hết hạn.

### Access token

Nếu client muốn lấy protected resources thì đầu tiến nó phải đạt được một access token. Hay nói cách khác, một client đầu tiên cần được cấp quyền từ user

Khi mà user cấp quyền một application, the authorization server trao client một access token. token này có mối quan hệ với user và được sử dụng khi mà muốn truy cập tới protected resources trên server, và đây là lý do tại sao chúng ta nói rằng một request đại diện user

Ví dụ:



* access\_token – định nghĩa một string, đại diện một authorization trên user, người mà đã cấp quyền cho request, phát tới client application
* expires\_in – nói với client application thời gian mà token này hợp lệ
* scope – (optional) thằng này xác định các phần của protected resource có thể được truy cập của thằng đang đại diện user
* state – (optional) được sử dụng cho chính bản thân client, thường là checks security. Giá trị của state cái mà client application gửi trong request sẽ giống với phần state phản hồi về
* refresh\_token – chứa string cái mà được sử dụng khi request một token mới trước khi nó bị hết hạn

### Refresh token

Một token luôn có hạn, nếu vô hạn thì bảo mật được cho là tồi. Khi mà một access token hết hạn, client sử dụng refresh token để nhận một access token mới, bởi việc liên hệ với authorization server và cung cấp data từ refresh\_token. Nếu thằng này data hợp lệ, the authorization server trả về một access token response mới tới client (nếu không có refresh token thì ta lại phải nhập username-password mới để yêu cầu token mới, với thằng này thì nó đã lưu dữ liệu mà hóa nên không cần nhập lại)

## Clients và endpoints

Nhìn lại diagram bước 3: client cung cấp thông tin tới authorization server để nhận access token, để làm điều này thì client cũng cần xác thực thành công.

## Client types và profiles

Khi đăng ký client, AuthServer không biết kiểu của client được đăng ký. Có 2 kiểu

* Confidential client – kiểu client application này có khả năng giữa các thông tin bí mật
* Public client – thằng này thì không có khả năng bảo mật, ví dụ như JavaScript application chạy trực tiếp trên browser hoặc moblie app nơi mà application logic là trên WebView

Clients chia ra làm 3 profiles

* Web application – được xem xét là client app bí mật và nó có nghĩa là nơi mà data được lưu trữ bảo vệ trên server sode và không thể truy cập trên public/client side
* User-agent-based application – thằng này là một application được download sau đó executed trên user-agent environment (ví dụ, trong web browser). Tất cả dữ liệu được downloaded, thằng này là một public application
* Native application – là một public client. Application này được cài trên thiết bị

## Endpoints

Một endpoint là một HTTP URL string cái mà định nghĩa địa chỉ sẽ được sử dụng trong request bởi một entity có khả năng requests

OAuth 2.0 có 3 endpoints quan trọng

2 trong số chúng là server endpoints:

* Authorization endpoint – client sử dụng endpoint này để được cấp quyền từ resource owner. Nếu thành công, client đạt được một authorization grant
* Token endpoint – client sử dụng để cung cấp authorization grant và nhận access token trả về

Và một endpoint client

* Redirection (callback) endpoint – authorization server sử dụng thằng này để trả về data với cac thông tin xác thực

Server endpoint thì thường được xác định trong code của client application, và client endpoint thì được xác định khi đăng ký client với authorization server

## Access scope

Scope là một tham số được sử dụng khi client tạo một request tới authorization hoặc token endpoints

Với parameternayf client có thể chỉ định các phần mà nó muốn access. Với scopes client limit nó bởi resource thì đây là trường hợp bảo vệ tốt. Khi request được xử lý, the authorization server validate scope được yêu cầu, và nếu nó không hợp lệ, client application không nhận được một access token (tức là quyền truy cập vào, ví như điện thoại thì có các quyền truy cập vào ảnh ọt các thứ)

# IdentityServer4 Integration with ASP.NET Core

OAuth 2 là một giao thức tiêu chuẩn cho authorzation. Nó ủy thác việc xác thực user tới một service, nơi mà chứa các thông tin về account và cấp cho ứng dụng bên thứ ba có thể truy cập vào account đó (ví dụ như google là một service, và nó cho phép ứng dụng bên ngoài có thể truy cập vào tk trên google thông qua service đó). OAuth là tên viết tắt của Open standard for Authorization.

Với authorization chúng ta có thể access tới một endpoint. Nhưng nếu chúng ta muốn thêm authentication trong process, chúng ta cần refer tới OpenId Connect. OpenID Connect là hoàn thiện của OAuth 2 với việc thêm phần authentication. Nó đơn giản là một identity layer bên trên OAuth2, nó cho phép các client kiểm chứng danh tính của họ sau khi học thực hiện xác minh trên authorization server. Chúng ta có thể trích xuất nhiều hơn thông tin về end-user bởi việc sử dụng OpenID Connect

## Proprietary auth with .NET Core

Sử dụng ASP.NET Core Identity system để thêm phần authentication và authorization tới application. Bắt đầu với .NET Core 3.0, ASP.NET Core Identity đã được tích hợp với một cơ chế API authorization mới để xử lý authentication trong SPAs: tính năng mới này dựa trên IdentityServer, một phần của OIDC và OAuth 2.0 middleware.

Với ASP.NET Core Identity, chúng ta có thể dễ dàng thực thi một cơ chế login, cơ chế này cho phép các users của chúng ta tạo một account và log in với username và password. Ở phía bên trên của thằng này, chúng ta có thể cho chúng khả năng sử dụng một external login provider.

* User – Các users đăng ký với application
* Role – Các roles cái mà chúng ta có thể gán tới user
* UserClaim – Các claims mà user sở hữu
* UserToken – Các token mà user đã phát tới client
* UserLogin – Các account có thể đăng nhập vào user (fb, google, twitter,..)
* RoleClaim – Bảng chung thể hiện quan hệ many-many của Role và Claim, mỗi role có thể có nhiều claim
* UserRole – Bảng chung thể hiện quan hệ many-many của User và Role, mỗi user có thể có nhiều role

## Identity và Access Control

Identity như cái tên đã giợi ý, nó có vài attributes cái mà chúng ta có thể sử dụng để đại diện một người, nhóm, ứng dụng hoặc thiết bị. Access control refer tới một kỹ thuật bảo mật được sử dụng để điều chỉnh người hoặc cái gì có thể access tới resource

## Authentication và Authorization

Authentication nghĩa là một quá trình xác định một user là ai, họ yêu cầu cái gì. Khi mà đã xác thực, authorization xác định tài nguyên mà user có thể truy cập hoặc không

## OpenID Connect và OAuth

OpenID Connect là layer cho việc identity và authentication đơn giản, xây dựng phía bên trên của OAuth, cho phép ứng dụng xác minh danh tính của end-users. OAuth cung cấp bảo mật, ủy quyền truy cập có nghĩa rằng một client có thể truy cập vào server tài nguyên mà không cần user phải luôn luôn chia sẻ chứng chỉ của họ với application (client) (có nghĩa là không cần cứ phải liên tục mà đăng nhập)

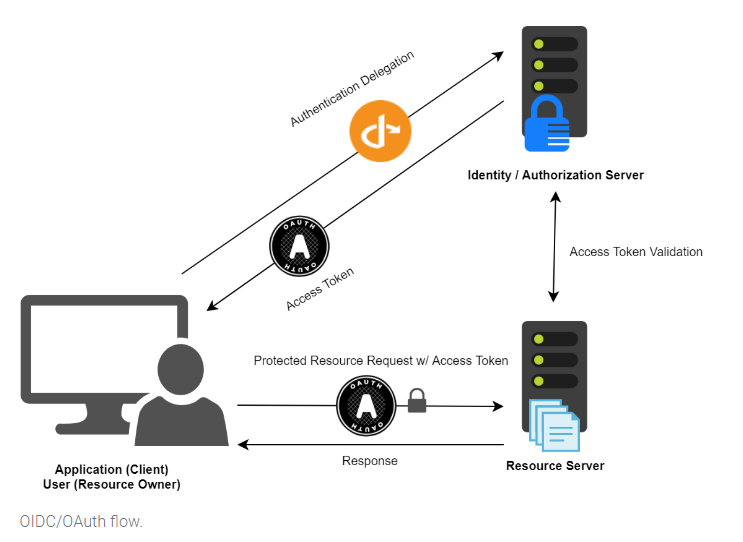


Diagram trên là như thế nào

1. The user bắt đầu request xác thực với authorization server
2. Nếu các chúng chỉ là hợp lệ, client nhận được một access token
3. access token sẽ được đính kèm cùng với các request tiếp theo tới resource server
4. authorization server kiểm tra access token, nếu thành công thì tài nguyên được bảo vệ sẽ được cấp và gửi lại client

Việc đăng nhập các thứ bây giờ sẽ triển khai riêng trên server, client sẽ đơn giản chỉ là lấy thông tin mà thôi dựa vào visa mà server cho về mà thôi. Có nghĩa công việc xác thực cấp quyền giờ được xử lý bởi nguyên 1 service

# IdentityServer4

## Protecting an API using Client Credentials

Trong kịch bản này, ta định nghĩa một API và một Client. The client sẽ request một access token từ IdentityServer bằng việc sử dụng client ID và secret key của nó, sau đó sử dụng token thu được từ Identity Server để truy cập vào API

### Tạo Identity Server

Trước tiên nói về Scope:

scope được sử dụng bởi client để xác định trong request kiểu truy cập nào được cấp tới họ trên danh nghĩa resource owner, và được sử dụng bởi authorization servers trong phản hổi để xác nhận.

Khi một access token được phát tới client, the scope đã xác định trong đó, định nghĩa quyền truy cập cho thành phần token này. Nó định nghĩa tài nguyên và API trên tài nguyên, client có thể truy cập và thực thi nó với token đã trao.