**Memahami OAuth 2.0 (API Security)[[1]](#footnote-1)**

**memahami Roles & Grant Type Flow**

[Tioreza Febrian](https://medium.com/@tiorzfk?source=post_page-----9376bc3a307b----------------------) [Jul 25, 2017](https://medium.com/codelabs-unikom/memahami-oauth-2-0-api-security-9376bc3a307b?source=post_page-----9376bc3a307b----------------------) · 8 min read

**Apa Itu OAuth ?**

Dikutip dari [wikipedia](https://id.wikipedia.org/wiki/OAuth) OAuth (O*pen Authorization*) itu adalah

Suatu [protokol](https://id.wikipedia.org/wiki/Protokol_(komputer)) terbuka yang memungkinkan pengguna untuk berbagi sumber pribadi mereka (mis. [foto](https://id.wikipedia.org/wiki/Foto), [video](https://id.wikipedia.org/wiki/Video), [daftar alamat](https://id.wikipedia.org/w/index.php?title=Daftar_alamat&action=edit&redlink=1)) yang disimpan di suatu [situs web](https://id.wikipedia.org/wiki/Situs_web) dengan situs lain tanpa perlu menyerahkan nama pengguna dan [kata sandi](https://id.wikipedia.org/wiki/Kata_sandi) mereka.

Ketika kita membuat *Third party apps* dan salah satu fitur di dalamnya yaitu mengakses lagu di Spotify misalnya. Resource Owner atau user akan memasukan username dan password spotify untuk kemudian mengakses lagunya. OAuth server akan menjembatani interaksi antara Resource Owner (user) dan Client (*Third party apps*). OAuth Server yang akan melakukan proses *Authentication* dan *Authorization*. Client akan mendapatkan Access Token dari OAuth Server jika proses Authentication dan Authorization berhasil. Client akan mengakses *Resources* melalui API yang disediakan *Resource Server* (Spotify) dengan menggunakan *Access Token*. Akan dijelaskan lebih jelas pada bagian Grant Type.

**OAuth Roles**

Disini kita akan mengenal beberapa peran atau aktor yang memerankan proses OAuth. Sama seperti aktor pada film yang memerankan peran secara spesifik. Pada OAuth pun terdapat aktor yang memerankan peran masing-masing sehingga proses OAuth dapat berjalan dengan baik.

**Resource Owner**

Resource Owner atau user adalah orang yang memiliki resources. Resource owner dapat mengakses data secara langsung ke twitter (contoh) dengan cara login pada twitter.com , namun dalam konteks OAuth bukan akses secara langsung yang dimaksud, tetapi secara tidak langsung dimana user mengakses data melalui *Third party apps.*

**OAuth Server / OAuth Provider**

Terdapat 3 Komponen di dalam OAuth Provider.

* Authentication Component

Komponen ini yang menyediakan login page yang ditampilkan ke user.

* Consent Component

Komponen ini akan muncul setelah proses authentication component selesai. Komponen ini akan memunculkan page lain yang berisi informasi persetujuan bahwa *Third party apps* akan mengakses beberapa informasi dari user.

* Token Management

Komponen ini bertugas untuk menjaga token-token yang sudah diberikan oleh OAuth Provider yang tersimpan pada database. Komponen ini juga bertugas memvalidasi token yang di request oleh client.

**Client (Third Party Apps)**

Client adalah aplikasi pihak ketiga yang akan mengakses data dari *Resource Server*. Client dapat berupa Web Apps, Mobile Apps, dll.

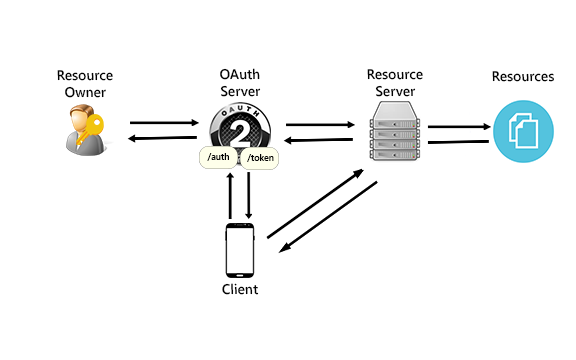
**Resource Provider / Resource Server**

*Resource server* ibarat rumah bagi resources (data) yang mana bertugas melindungi resources. *Resource Owner* dapat mengakses *resources* melalui *Resource Server* yang diakses dengan menggunakan HTTP Protokol dan disediakan dalam bentuk Web API. Web API harus memastikan bahwa hanya user yang ter-*authentication* dan *authorization* yang dapat mengakeses data. Client harus memberikan *access token* pada HTTP Header ketika ingin merequest suatu data. Yang harus pertama kali Web API lakukan di setiap request yang masuk adalah memeriksa validitas *access token* ke *OAuth Server*. Jika *access token* yang dikirim valid maka client berhak mendapatkan data.

**Grant Types**

Grant Type membuat proses OAuth 2.0 sangat flexible. Grant Types merupakan proses pemberian akses ke *resources (sumber data)* yang dilindungi dengan berbagai cara dan keamanan data yang berbeda.

**Authorization Code Grant**

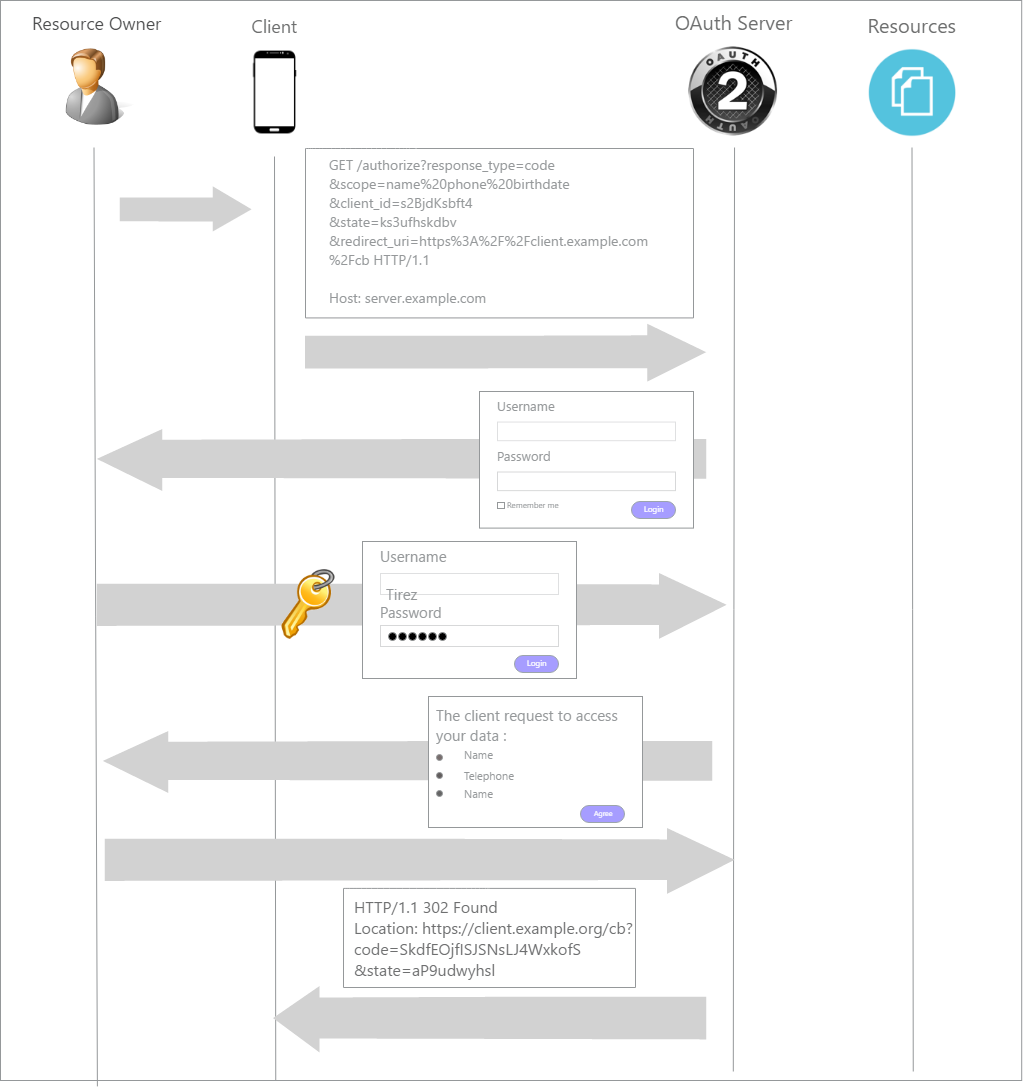


Tirez Autorization Code Grant Type

Jenis grant type *Authorization Code* ini tidak beda jauh dengan OAuth 1.0 dengan enkripsi yang lebih sedikit. Jenis grant type ini paling aman diantara jenis grant type yang lain. Ada beberapa tahapan dalam proses Authorization Code Grant Type.

1. **Authorization Code**

Pada tahap ini Client akan meminta authorization code menggunakan endpoint yang bernama ’ */auth’.* Endpoint authorization tidak semua ‘*/auth’* , tergantung resource server yang menyediakan endpoint nya.



Proses yang pertama ini melibatkan resource owner, client, dan OAuth Server. Tujuan dari proses yang pertama ini yaitu mendapatkan authorization code yang akan digunakan untuk mendapatkan *access token*. Proses ini dimulai ketika *Resource Owner* menggunakan *Third Party Apps* untuk melakukan sesuatu (meminta data misalnya). Lalu Client akan pergi ke OAuth Server dengan memanggil endpoint */authorize.* Terdapat beberapa parameter yang dibawa ketika client request endpoint */authorize* :

* **response\_type**

REQUIRED. Diisi ‘code’ untuk standar OAuth2 authorization flow.

* **client\_id**

REQUIRED. Client id merupakan identitas dari *Third Party Apps.* Client id didapatkan setelah mendaftarkan *Third Party Apps*.

* **scope**

OPTIONAL. Scope digunakan untuk mendefinisikan informasi apa saja yang akan diminta. Dalam gambar diatas kita ingin meminta data nama, phone, dan birthday.

* **state**

RECOMMENDED. Random String, Di generate oleh client. Digunakan untuk mencegah serangan [CSRF](https://en.wikipedia.org/wiki/Cross-site_request_forgery) (baca : sea-surf) dan digunakan oleh klien untuk me-maintain antara request dan callback.

* **redirect\_uri**

Digunakan oleh OAuth Server untuk mengembalikan hasil. Redirect URI yang dikirim *Third Party Apps* ke *OAuth Server* harus sama dengan Redirect URI yang didaftarkan ketika *Third Party Apps* mendapatkan Client Id.

Jika parameter yang dimasukan valid, maka *OAuth Server* akan me-redirect ke sebuah halaman login seperti gambar diatas.

*Resource Owner* akan memasukan username dan password yang akan dikirimkan kembali ke *OAuth Server*.

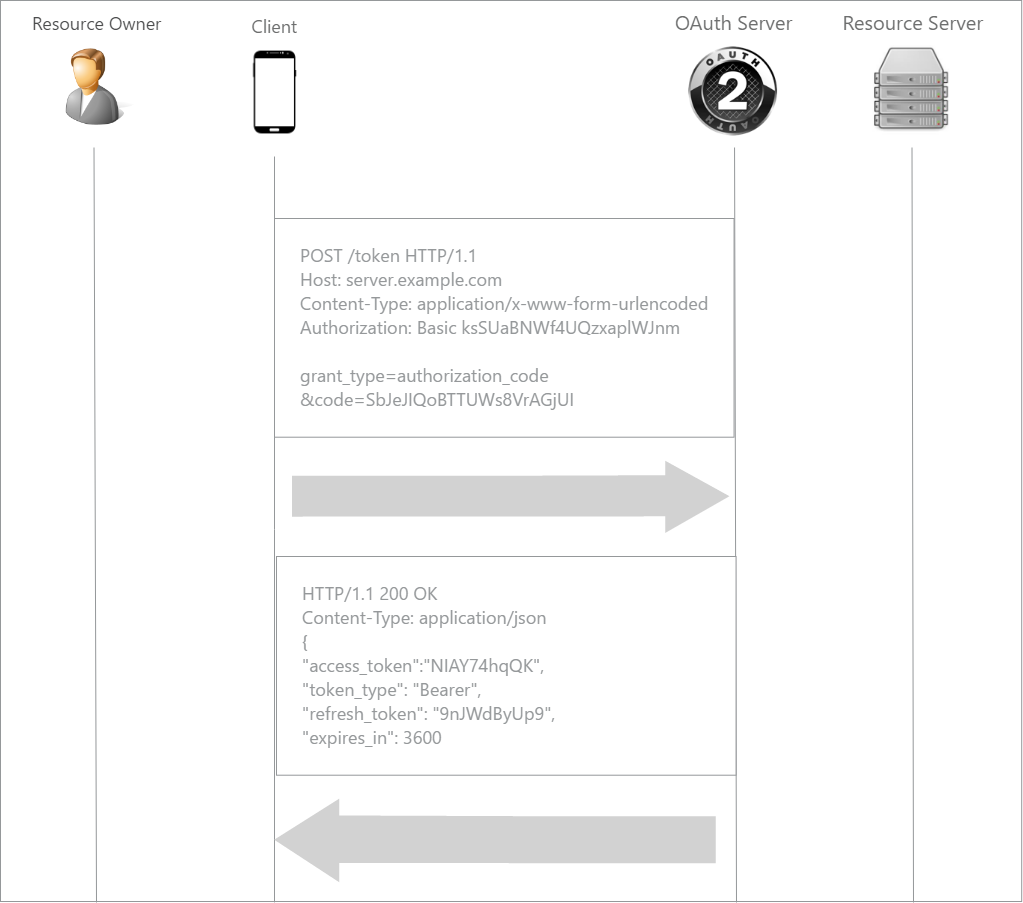
*OAuth Server* akan memeriksa apakah username dan password yang dimasukan valid.

Jika valid maka *OAuth Server* akan mengirimkan kembali sebuah halaman persetujuan apakah *Resource Owner* mengizinkan datanya diakses oleh *Third Party Apps.* Data tersebut sesuai dengan data yang di request oleh *Third Party Apps* pada parameter *scopes.*

Jika *Resource Owner* setuju maka OAuth Server akan mengirimkan sebuah data ke client melalui HTTP Redirect. Data tersebut berupa parameter code dan state. Code akan digunakan client untuk ditukarkan menjadi token, dan state yang diterima merupakan state yang sama dengan state yang dikirim oleh client.

**2. Get the Token**

Sampai saat ini kita sudah mempunyai Authorization Code. Selanjutnya kita memerlukan *Access Token* untuk request suatu *Resources*.



Tirez Authorization Code Flow

untuk mendapatkan access token, kita harus request satu endpoint lagi yang bernama */token* dengan method **POST**.

Jenis Authorization yang digunakan untuk request token yaitu **HTTP Basic** yang mana terdiri dari username dan password yang dipisahkan dengan titik dua (colon) dan di encode menggunakan base64.

Data username dan password disini merupakan data yang dimiliki oleh *Client (Third Party Apps)* bukan *Resource Owner.* Username yang dimiliki *Third Party Apps* yaitu sama dengan client id yang kita lihat sebelumnya dan Password yang dimiliki Third Party Apps merupakan Client Secret yang kita dapatkan bersamaan dengan Client Id.

Dengan mencantumkan data tersebut maka OAuth Server mengetahui *Third Party Apps* yang mana yang sedang berinteraksi dengan dia.

Parameter Grant Type digunakan untuk me-spesifikasi kan jenis grant type yang kita gunakan. Karena sekarang kita menggunakan Authorization Code Grant maka diisi dengan ‘authorization\_code’.

Parameter code merupakan Authorization Code yang kita dapatkan pada tahap pertama.

OAuth Server akan memeriksa validitas data yang dikirim oleh client. Jika aman maka OAuth Server akan mengirimkan kembali hasilnya ke Client seperti gambar diatas.

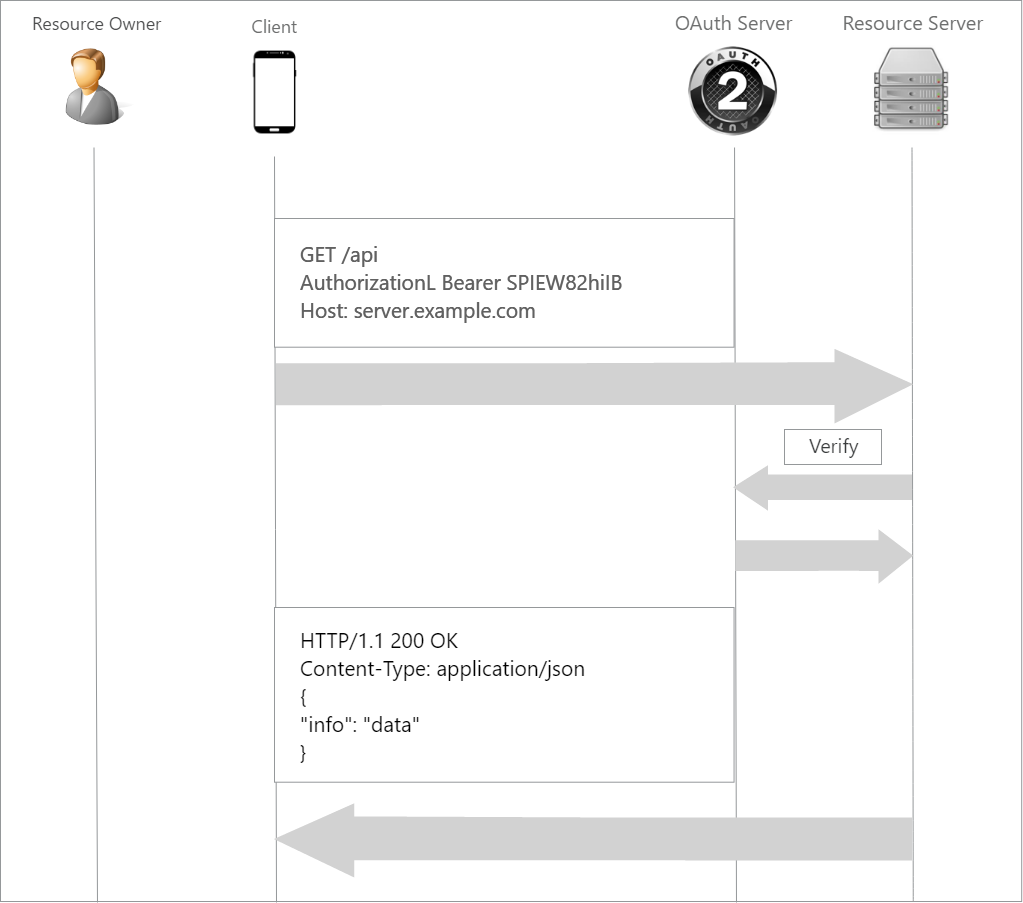
**access\_token** digunakan untuk mengakses Resources menggunakan **HTTP Bearer**.

**refresh\_token** akan digunakan untuk me-generate Access Token yang baru, setelah access token kadaluarsa. Masa kadaluarsa refresh token akan lebih lama dibanding access token.

**expires\_in** merupakan masa kadaluarsa dari access\_token dalam satuan detik.

**3. Use the Token to Access a Resource**

Dalam tahap ini merupakan tahap dimana client mengakses Resources menggunakan token.



Tirez Authorization Code Flow

Client akan mengakses data dengan request ke salah satu endpoint yang disediakan **Resource Server** sesuai dengan data yang dibutuhkan.

Pada gambar diatas kita akan mencoba request suatu data dengan endpoint **/api** dan dengan method **GET**. Dikirim menggunakan Bearer Authorization Header dengan mencantumkan access\_token nya.

Resource Server mendapatkan token yang di request oleh client. Resource Server tidak tahu apakah token yang dia dapat valid atau tidak, maka resource server akan mengirimkan token tersebut ke OAuth Server.

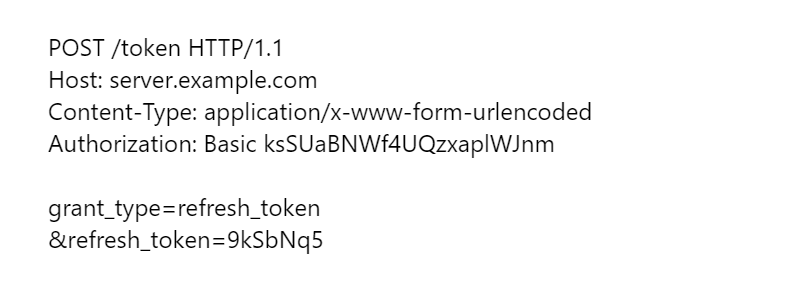
OAuth Server akan memeriksa token yang dikirim *Resource Server* apakah token tersebut valid, masih berlaku atau tidak.

OAuth Server akan mengirimkan hasil nya kembali ke *Resource Server* berupa informasi bahwa token tersebut valid atau tidak.

Jika valid maka *Resource Server* akan mengirimkan data berupa informasi yang dibutuhkan sesuai dengan permintaan client.

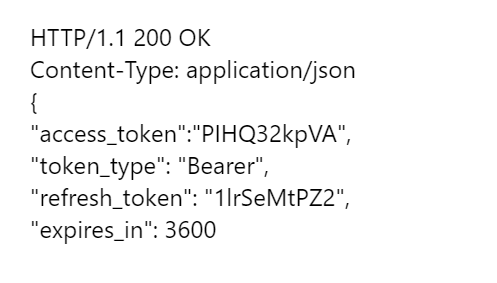
**Authorization Code Grant: Refresh Token**

Seperti yang kita tahu sebelumnya access token mempunyai batas kadaluarsa, maka dalam waktu tertentu access token perlu di refresh. Dengan refresh token kita tidak perlu melakukan authorization kembali untuk mendapatkan access\_token yang baru.



Tirez Refresh Token

Untuk mendapatkan access token yang baru client perlu request endpoint */token* dengan parameter grant\_type diisi dengan refresh\_token dan parameter refresh\_token diisi dengan refresh token yang didapatkan ketika berhasil melakukan Authorize pada tahap sebelumnya.

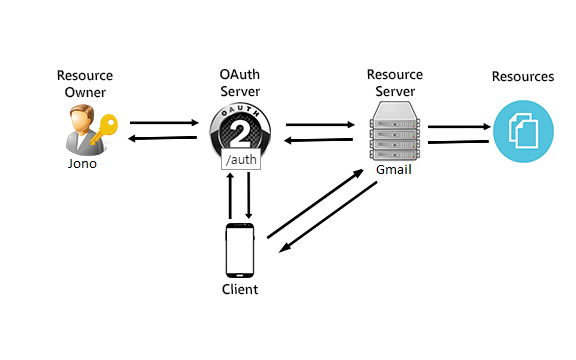


Tirez Refresh Token

Jika berhasil maka OAuth Server akan mengirimkan kembali access token, refresh token baru beserta yang lainnya.

**Implicit Grant**

Implicit Grant merupakan penyederhanaan dari Authorization Code Grant.

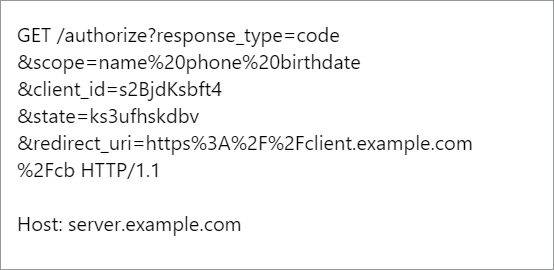


Tirez Implicit Flow

Implicit Grant tidak menggunakan endpoint */token* seperti yang terlihat pada gambar diatas dan juga tidak mengirimkan refresh token.

Grant type ini biasanya digunakan untuk client yang kemungkinan tidak dapat menyimpan refresh token, client id, dan client secret dengan aman. Contohnya client-side javascript yang berjalan pada browser *resource owner*.

Langkah pertama dalam flow ini sama dengan authorization code flow yaitu request authorization endpoint seperti pada gambar dibawah ini.



Tirez Implicit Flow

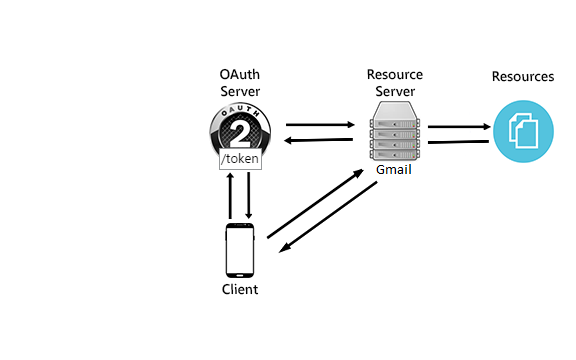
Setelah semua langkah berhasil OAuth Server akan langsung memberikan access token, token type dan state , tanpa memberikan Authorization Code terlebih dahulu seperti yang dilakukan di Authorization Code Flow.



Tirez Implicit Flow

**Client Credentials Grant**

Client Credentials hanya menggunakan endpoint */token,* tidak menggunakan endpoint */auth,* tidak memerlukan *resource owner* untuk meng otentikasi dirinya dan tidak ada refresh token*.*



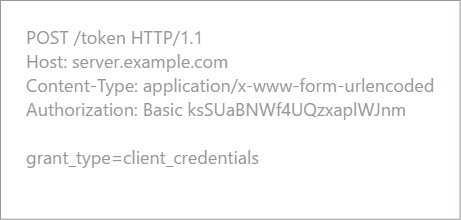
Tirez Client Credentials Grant

Karena flow ini tidak terdapat Authorization maka flow ini tidak dapat digunakan untuk mengakses atau mengelola data pribadi pengguna.

Client (Third Party Apps) memiliki client id dan client secret. Untuk mendapatkan access token, client harus request ke endpoint /token dengan method POST.

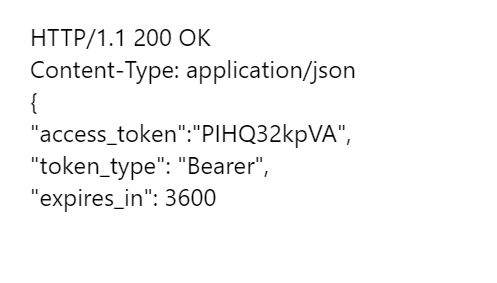
Jenis Authorization yang digunakan adalah **HTTP Basic** yang mana terdiri dari client id dan client secret yang dipisahkan dengan titik dua (colon) dan di encode menggunakan base64.

Parameter yang dibawa yaitu grant type yang berisi client\_credentials.

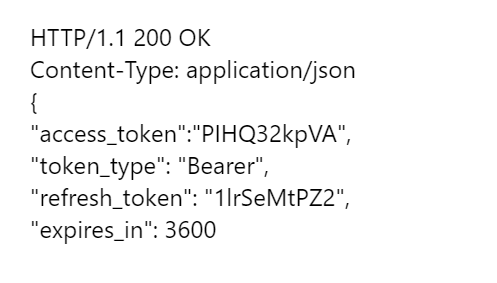


Tirez Client Credential Flow

Jika berhasil maka OAuth Server akan merespon data seperti berikut.

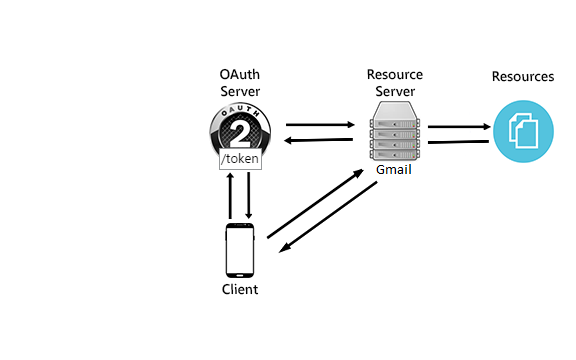


Tirez Client Credential Flow



Tirez Refresh Token

**Resource Owner Password Credential Grant**



Tirez Client Credentials Grant

Pada jenis grant ini resource owner akan memasukan username dan password nya secara langsung ke client (Third Party Apps).

Grant type ini biasanya digunakan jika aplikasi yang dibangun merupakan aplikasi pribadi atau pembuat aplikasi berasal dari perusahaan yang sama.

Client mungkin merupakan official mobile apps dari *resource server*, seperti official mobile facebook app yang mana facebook sebagai *resource server*.

Grant Type ini membawa risiko lebih tinggi daripada Grant Type lainnya karena ini mempertahankan konsep [password anti pattern](http://api-university.com/blog/the-password-anti-pattern/" \t "_blank) yang ingin dihindari oleh protokol ini. Client bisa menyalahgunakan password *resource owner* dan client bisa saja secara tidak sengaja memberikan data *resource owner* (mis. Melalui file log atau catatan lainnya disimpan oleh Client).

Alur dari grant type ini adalah seperti berikut.

* Resource owner memasukan username dan password ke client (Third Party Apps).
* Client meminta access token ke OAuth Server menggunakan endpoint */token* dengan memasukan credentials yang didapatkan  
  dari *resource owner*.
* OAuth Server akan melakukan authentication dan validasi credentials resource owner, jika valid, maka akan menampilkan access token.

Akhir kata semoga bermanfaat dan semoga menambah pemahaman seputar OAuth 2.0.

Cheers.

1. https://medium.com/codelabs-unikom/memahami-oauth-2-0-api-security-9376bc3a307b [↑](#footnote-ref-1)