## Cas d'usage

Voici deux cas d'usage de OAuth qui permet de comprendre rapidement à quoi cela sert:

J'ai un compte mail Yahoo et un autre compte sur Gmail.

*Je souhaite récuperer dans mon compte Yahoo tous mes contacts de mon compte Gmail*

A partir de Yahoo, je demande un *import* de contacts depuis mon compte Gmail. Il se met alors en place un protocole de délégation d'accès basé sur Oauth.

L'application Yahoo me demande la permission d'accéder à mes contacts du compte Gmail pour lequel je dois au préalable m'authentifier.

Si j'étais déjà authentifié sur Gmail, dans un autre onglet par exemple, seul apparaît la demande d'accèder aux contacts

Remarque importante: C'est moi, en tant qu'utilisateur qui m'authentifie. A aucun moment je ne fournis à Yahoo mes *credentials* Gmail. Je luis délègue seulement un droit d'accès à mes données personnelles.

On voir ainsi que OAuth permet à un utilisateur de déléguer l'accès à ses données personnelles à une tierce partie que l'on appelle le Client (dans l'exemple ci-dessus, il s'agit de l'application Yahoo Mail)

Pour que Yahoo puisse interagir avec les ressources détenues par Gmail, il aura fallu que le compagnie Yahoo inscrive sont application auprès des APIs de Google permettant de manipuler des objets de type *Contact*. A l'issue de cet enregistrement,

Yahoo obtient un id de service ainsi qu'un clef secrète qui sera utilisée lors de l'échange entre Yahoo et Gmail.

### Autre cas d'usage théorique avec Oracle Documents Cloud Service (ODCS)

Je veux créer un nouveau service destiné à présenter les documents de Oracle Documents Cloud Service (ODCS) dans une page HTML, et combinés avec des informations provenant d'un SI; L'objectif est d'obtenir un affichage contextuel de documents en fonction d'un objet métier (contrat, projet, opportunité …).

Je pourrais avoir recours à un framework Javascript et utiliser les APIs REST de ODCS, mais cela n'est pas possible actuellement car ODCS ne supporte pas le *Cross Scripting*.

Je suis donc conduit à effectuer mon traitement sur un serveur web quelconque. Je décide de développer l'application en Java. Celle-ci se connecte au SI et également à ODCS pour effectuer les appels REST. Ici, l'application Java doit fournir un token pour la Basic authentication qui est exigée par ODCS. Cela signifie donc que l'usager doit fournir son compte et mot de passe à l'application spécifique, et c'est là que c'est gênant puisqu'on oblige l'utilisateur à *dévoiler,* en quelques sorte*,* ses *credentials* à un tiers (l'application dans notre cas).

Si ODCS supportait OAuth, alors l'utilisateur s'authentifierait auprès de ODCS et ODCS fournirait un jeton temporaire d'accès à l'application spécifique.

### Présentation

Excellent article en Français publié par CA pour son produit CA Layer 7

http://www.ca.com/fr/~/media/Files/whitepapers/a-how-to-guide-to-oauth-and-api-security-final-fra.PDF

Autres liens:

- Acces console developpeur Google
- Article sur OAuth2
- Sample acces Google Drive
- Le Blog d'Eran Hammer (un des concepteurs de OAuth qui a démissionné en 2012 et qui n'a pas la langue dans sa poche)

## Définitions

Il est intéressant de confronter les définitions successives de OAuth v1 et OAuth v2:

> V1
> OAuth provides a method for clients to access server resources onbehalf of a resource owner (such as a different client or an end-user).
> It also provides a process for end-users to authorize third-party access to their server resources without sharing their credentials (typically, a username and password pair),using user-agent redirections.

> V2
> The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner
> by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf.

This specification replaces and obsoletes the OAuth 1.0 protocol described in RFC 5849.

## Author

Patrick

GPM Factory