



DB2P datatransfer  
Context & Onboarding

## › Contenu

---

|   |    |
|---|----|
| 1. Context DataTransfer .....   | 3  |
| 2. Procédure .....  | 4  |
| 3. Guide d'intégration Oauth2 .....   | 5  |
| 3.1 Présentation du protocole OAuth 2.0 .....                                     | 5  |
| 3.1.1 Lexique .....   | 5  |
| 3.1.2 Acteurs . . . ..  | 5  |
| 3.1.3 Etapes du processus d'autorisation maar ook een rol voor de werkgever ..... | 6  |
| 3.2 Les flux OAuth 2.0.....   | 9  |
| 3.2.1 Authorization Code Grant Flow .....   | 9  |
| 3.2.2 Implicit Grant Flow .....   | 9  |
| 3.2.3 Client Credentials .....  | 10 |
| 3.2.4 Quel flux OAuth choisir ?.....  | 10 |

## › 1. Context DataTransfer

---

- Le projet DB2P DataTransfer consiste en un transfert des données de pension complémentaire d'un citoyen vers une instance tiers à l'**initiative du citoyen**.
- Les données transférées sont quasiment les mêmes données que celles présentées au citoyen dans l'application MyPension/Ma pension complémentaire.
- L'instance tiers doit demander une autorisation au Comité de Sécurité de l'Information (CSI) (<https://www.ksz-bcss.fgov.be/fr/protection-des-donnees/comite-de-securite-de-linformation-csi>)
- L'instance tiers doit prévoir que le citoyen (ci-après "ressource owner") puisse se connecter via une authentification substantielle ou élevée dans leur application web/app via le FAS (federal authentication service) ([https://dt.bosa.be/fr/identification\\_et\\_securisation/federal\\_authentication\\_service](https://dt.bosa.be/fr/identification_et_securisation/federal_authentication_service)).
- Ce mandat prendra la forme d'un Token (Authorization server / OAuth).
- Le mandat est limité dans le temps. Il y a deux sortes de mandat: prospectif (avec une durée de 1 heure) ou basé sur une relation contractuelle (avec une durée de 2 ans). Un mandat est toujours réutilisable durant la période de validité et peut toujours être renouvelé ensuite.
  - › **Le mandat prospectif** (de 1 heure) correspond à des situations dans lesquelles le citoyen donne une seule fois son accord pour un transfert de données.
    - (Exemple 1: les données sont visualisées 1 seule fois dans l'application d'un tiers.)
    - (Exemple 2: les données sont chargées une fois dans l'application d'un tiers à des fins de calcul ou de conseils.)
  - › L'autre **mandat a une durée de validité de 2 ans**. Ce mandat correspond à des situations dans lesquelles il y a une relation longue durée entre le citoyen et le tiers et au travers de laquelle une actualisation régulière des données est nécessaire.
    - (Exemple 1: le citoyen souhaite voir ses données de pension complémentaire actualisées à chaque fois qu'il utilise l'application d'un tiers.)
    - (Exemple 2: le citoyen veut que le tiers suive ses données et lui donne des conseils sur mesure chaque fois que quelque chose change dans sa situation.)
- ! Sigedis ne contrôle que la présence d'un mandat encore valide et n'exerce aucun contrôle à priori sur le contexte d'utilisateur pour vérifier le respect des conditions d'utilisation restrictives que le citoyen aurait conclues avec le tiers.
- Nous conseillons à l'instance tiers d'utiliser, dans ses écrans à destination du citoyen, les termes déjà utilisés et montrés au citoyen dans l'application MyPension/Ma pension complémentaire (voir DataDictionary).

## › 2. Procédure

---

En dehors du fait que l'instance tiers doit demander une autorisation au Comité de Sécurité de l'Information (CSI) et prévoir la connection du citoyen via le FAS, l'instance tiers doit également passer par les étapes suivantes:

1. L'instance tiers doit introduire une demande pour obtenir le DB2P datatransfer à l'adresse email suivante : [DB2Pdatatransfer@sigedis.fgov.be](mailto:DB2Pdatatransfer@sigedis.fgov.be). La demande doit contenir le document onboarding complété (1 document par cas d'utilisation souhaité).

2. Une fois le document complété et renvoyé, Sigedis se chargera de la configuration et préviendra l'instance tiers une fois que tout est prêt.

3. L'URL de la ressource (L'URL pour accéder aux données de pension complémentaire) est d'ores et déjà disponible dans le document « cookbook » joint et dans le document « onboarding » à remplir. Celle-ci pourra être utilisée dès que la configuration aura eu lieu.

4. Il est à noter qu'un environnement Sandbox (Acceptation) est prévu en plus de l'environnement de production.

C'est dans cet environnement que l'instance tiers peut tester le processus complet (à partir du moment où le citoyen se connecte et donne son autorisation, jusqu'au moment où l'instance tiers reçoit les données de pension complémentaire).

Pour préparer au mieux ces tests, l'instance tiers fournira à Sigedis le numéro d'identification national (niss) des personnes en interne chez l'instance tiers qui, pour les tests, se connecteront avec leur carte d'identité.

Sigedis pourra ainsi créer des données fictives dans sa base de données en Acceptation et fournira un excel reprenant les résultats auxquels l'instance tiers peut s'attendre lors de ses tests.

## › 3. Guide d'intégration OAuth2

### ›› 3.1 Présentation du protocole OAuth 2.0

OAuth 2.0 est un protocole de sécurité permettant à une personne de déléguer ses droits d'accès à une ressource. Ce protocole permet à une application d'agir au nom de quelqu'un sans devoir fournir les informations secrètes de cette personne. Ce protocole est utilisé pour sécuriser les web services REST.

L'application cliente récupère auprès d'un serveur d'authentification un Access Token qui va lui permettre d'accéder à une ressource protégée au nom du propriétaire de la ressource.

Ce document définit la procédure à suivre afin de s'intégrer avec une application sécurisée via le serveur d'autorisation.

#### 3.1.1 Lexique

Ci-dessous se trouvent les termes spécifiques à OAuth que nous allons utiliser dans ce document.

**Resource** : est quelque chose dont on veut protéger l'accès et pour lequel le Resource owner peut choisir de déléguer les droits d'accès.

OAuth ne définit pas la nature de la ressource. Il est donc possible de construire un système de gestion des droits d'accès à des applications. Dans ce cas, la ressource est le droit d'accès.

**Token** : est un objet obtenu auprès de l'Authorization Server qui permet de prouver que le propriétaire de la ressource nous a confié l'accès à celle-ci. OAuth définit deux types de Token différents :

- › **Access Token** utilisé pour accéder à une ressource
- › **Refresh Token** permettant de renouveler un Access Token dans certains flux d'autorisation sans nécessiter la présence du Resource owner

**Scope** : Permet de définir les droits d'accès à la ressource délégués par le Resource owner. Par exemple, avec le scope "read", une personne peut accéder à la ressource mais ne peut pas la modifier.

#### 3.1.2 Acteurs

**Resource owner**: Le propriétaire de la ressource qui veut autoriser une application à agir en son nom. Généralement, celui-ci est une personne.

**Protected resource** : La ressource protégée auquel le propriétaire a accès. Celle-ci peut avoir différentes formes mais généralement c'est une web API qui peut avoir différents droits d'accès (lecture, écriture,...)

**Client** : Le client est l'application qui accède à la ressource protégée au nom du propriétaire de cette ressource. Cette application peut par exemple être une application située sur un serveur, une application javascript ou native.

**Authorization server** : Le serveur d'autorisation permet de fournir un Access Token que le client peut utiliser pour agir à la place du propriétaire de la ressource. Il va pour cela authentifier le propriétaire et vérifier son accord.

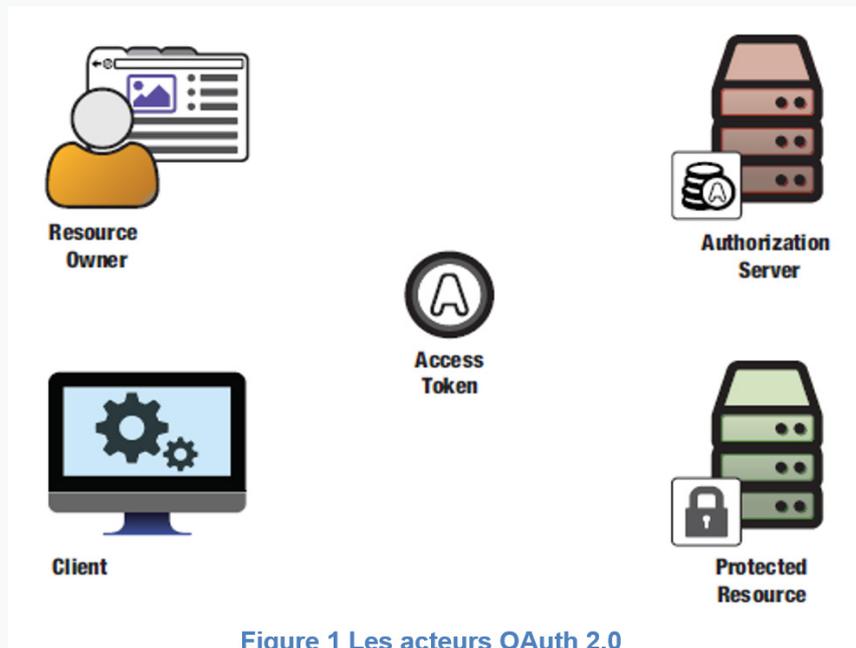


Figure 1 Les acteurs OAuth 2.0

### 3.1.3. Etapes du processus d'autorisation

Au départ du processus, le Resource owner contacte le Client (1) et lui signale qu'il aimerait que celui-ci agisse en son nom. Suite à cette demande, le Client contacte l'Authorization server (2) et indique qu'il veut pouvoir agir à la place du Ressource owner. Pour cela, le client met le Resource owner en contact avec l'Authorization server en transmettant différentes information (3) permettant d'identifier les droits demandés sur la ressource (via le scope) ainsi qu'une URL permettant à l'Authorization server de contacter le client une fois la demande accordée.

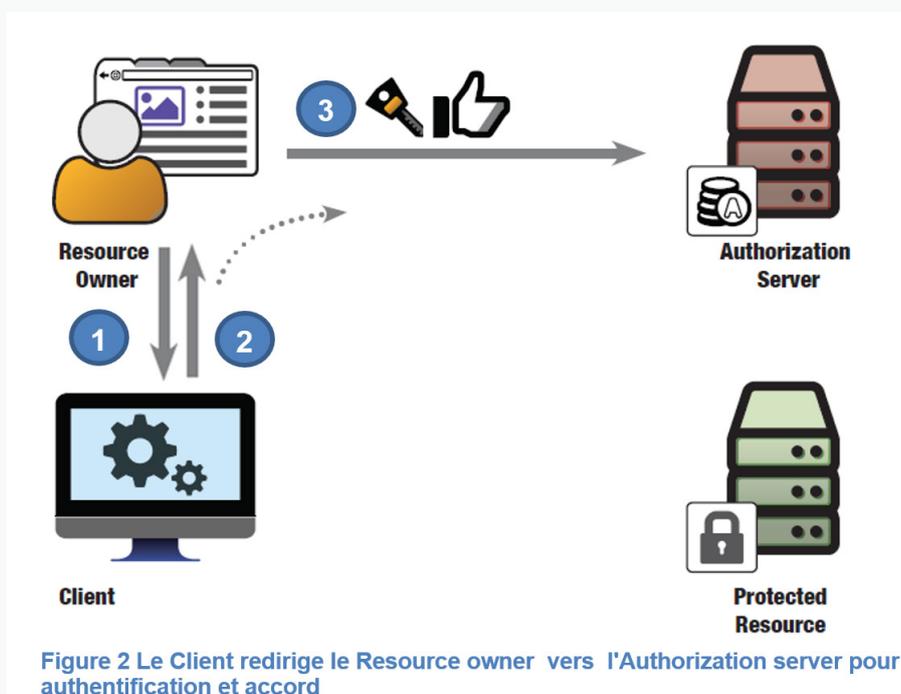


Figure 2 Le Client redirige le Resource owner vers l'Authorization server pour authentification et accord

Une fois le Ressource owner identifié et authentifié auprès de l'Authorization server, celui-ci demande l'approbation du Resource owner (1) en précisant les droits (scope) demandés par le client sur la ressource. Une fois l'approbation obtenue, l'Authorization server redirige le Resource owner vers le client (2) en utilisant l'URL de redirection obtenue précédemment. Via cette redirection, l'Authorization server transmet un code qui permettra au Client d'obtenir un Access Token.

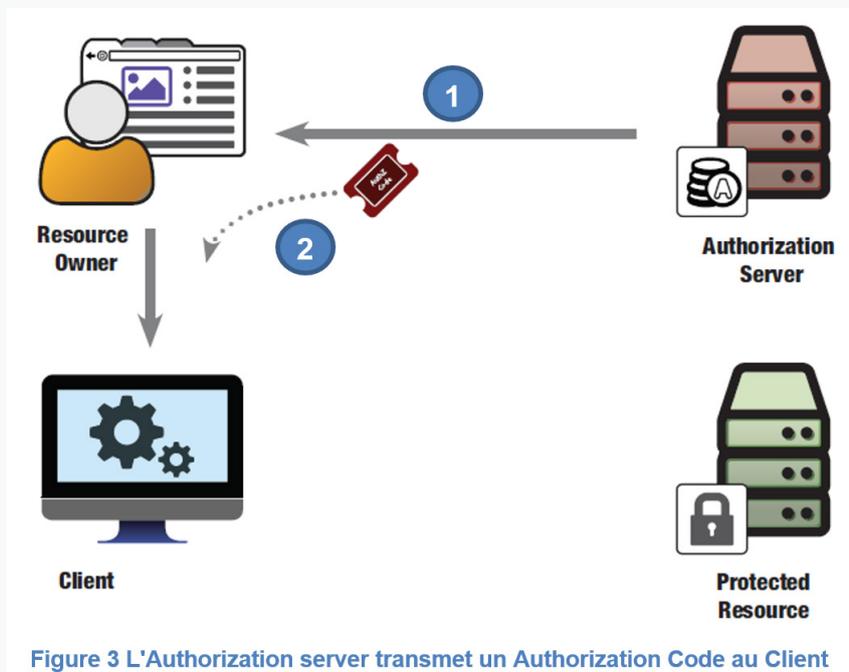


Figure 3 L'Authorization server transmet un Authorization Code au Client

Une fois en possession de l'Authorization Code, le Client contacte l'Authorization Server (1) et lui transmet son identifiant et secret permettant de l'authentifier ainsi que le code reçu précédemment. Sur base de ces informations, l'Authorization server retourne un Access Token (2).

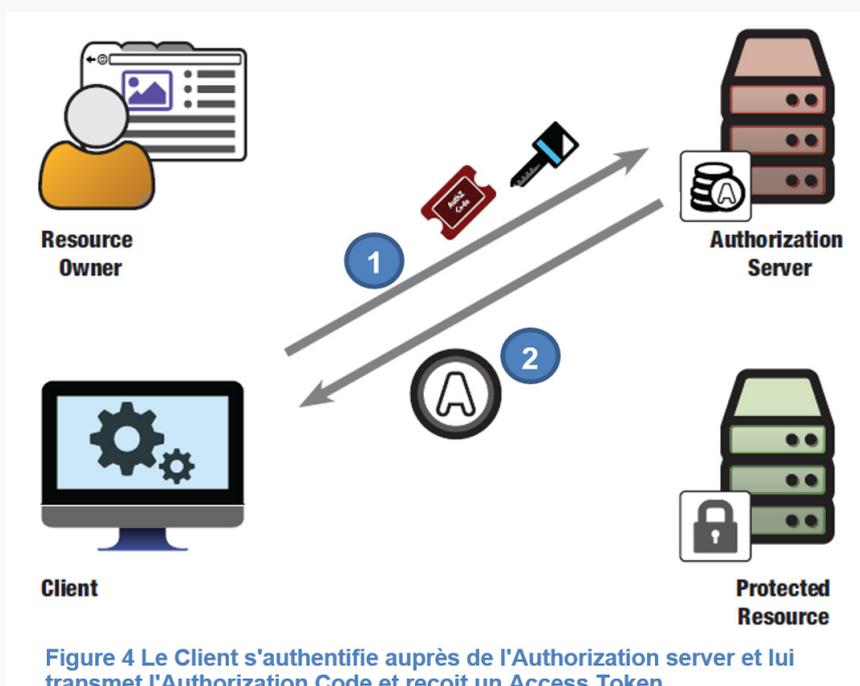


Figure 4 Le Client s'authentifie auprès de l'Authorization server et lui transmet l'Authorization Code et reçoit un Access Token

Une fois le Client en possession d'un Access Token et ceci tant que l'Access Token reste valide, il peut utiliser celui-ci pour accéder à la ressource. Pour ce faire, il va transmettre son identifiant, son secret et l'Access Token au Resource server (1). Celui-ci contacte l'Authorization server en transmettant ces informations (2) afin que celui-ci vérifie la légitimité de la demande. Une fois la demande approuvée (3) par l'Authorization server, le Resource server retourne au Client la ressource demandée.

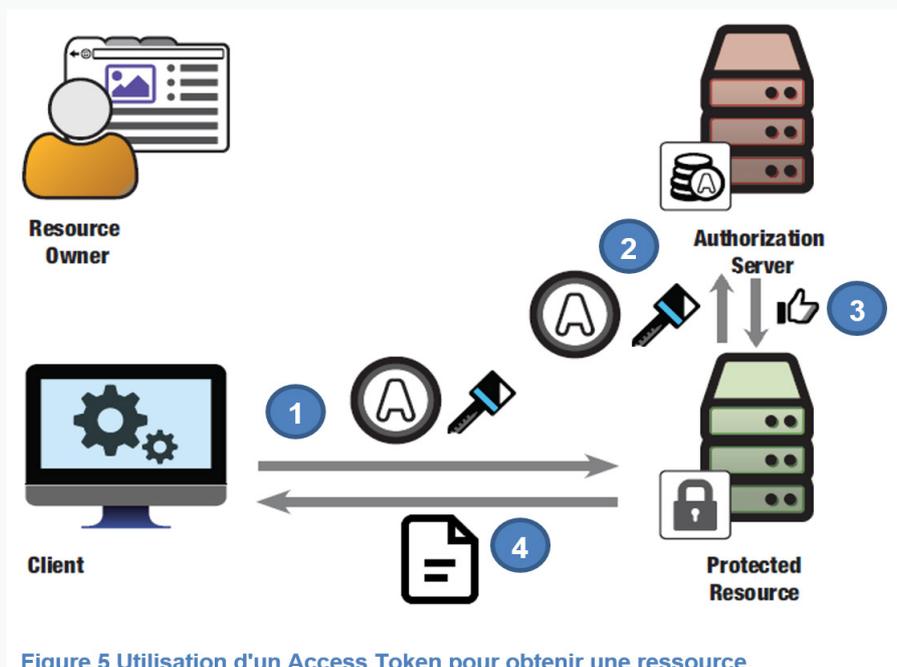


Figure 5 Utilisation d'un Access Token pour obtenir une ressource

## >> 3.2 Les flux OAuth 2.0

OAuth définit plusieurs flux d'autorisations à utiliser dans différentes situations. Chaque flux a un niveau de sécurité différent et doit être utilisé dans des situations particulières. Le flux le plus général et le plus sécurisé est celui de l'Authorization Code. Les autres sont des optimisations de celui-ci.

De plus, un Access Token a une durée de validité déterminée à sa création. Un Access Token peut donc devenir invalide pour deux raisons : il peut avoir expiré mais il peut aussi être révoqué par le Resource Owner. Comme chaque flux est différent, chacun a une manière différente de gérer le renouvellement des Access Token.

### 3.2.1 Authorization Code Grant Flow

Ce type d'autorisation est celui présenté en détail dans le chapitre précédent. Celui-ci requiert plusieurs échanges d'informations différents (Authorization code, Access Token, Refresh Token) afin de compartimenter un maximum les différentes étapes de manière à ce qu'aucune des parties impliquées n'ait accès à des informations dont elle n'a pas besoin.

Ce flux permet dans un premier temps d'obtenir un Access Token ainsi qu'un Refresh Token. Par la suite, ce flux permet aussi le renouvellement d'un Access Token au moyen du Refresh Token reçu. Un Refresh Token a une durée de vie plus longue et peut être fourni à l'Authorization server pour renouveler l'Access Token. Le Refresh Token n'étant valable qu'une seule fois, un nouveau Refresh Token est fourni lors de ce renouvellement. Cette méthode permet un renouvellement de l'Access Token sans nécessiter la présence du Resource Owner.

Pour plus d'informations concernant ce flux, vous pouvez consulter le cookbook aux pages 5 et 8.

### 3.2.2 Implicit Grant Flow

Le point fort du flux Authorization Code Grant est qu'il garde séparées les informations propres aux différents composants. Cependant, lorsque le Client est une application s'exécutant entièrement au sein d'une même page web (en Javascript), cette séparation est impossible. L'échange d'informations menant à l'obtention d'un Authorization Code est donc inutile. De plus, le Client n'ayant pas de moyen de protéger son secret, celui-ci devient aussi inutile.

De plus, lorsque le Client n'est pas un programme s'exécutant sur un serveur distant mais plutôt au sein d'une application Javascript ou une application native, il n'existe pas de moyen de stocker de manière réellement sécurisée un client secret ou un Refresh Token.

Pour ces deux raisons, l'utilisation d'un Authorization Code et d'un client secret n'est pas toujours possible. Dans ce cas, il faut utiliser le flux d'autorisation implicite.

Pour plus d'informations concernant ce flux, vous pouvez consulter le cookbook aux pages 6 et 14.

### 3.2.3 Client Credentials

Ce flux permet d'utiliser OAuth dans le cas où une application n'agit pas au nom d'une tierce partie mais en son nom propre. Dans ce scénario, le client est donc le Resource Owner et il n'y a pas besoin d'échanger un Authorization Code ni de donner son accord.

Il n'est pas nécessaire de fournir un Refresh Token lors de ce flux car le client étant toujours présent pour s'authentifier, le Refresh Token perd son utilité.

Pour plus d'informations concernant ce flux, vous pouvez consulter le cookbook aux pages 7 et 17.

### 3.2.4 Quel flux OAuth choisir ?

Le choix d'un flux d'autorisation dépend principalement du type de Client utilisé.

Dans la majorité des cas, le flux approprié est celui de l'Authorization code. Cependant, deux situations nécessitent des flux différents :

› **Implicit Grant Flow** : est nécessaire lorsque le Client ne se trouve pas dans un environnement sécurisé afin de ne pas donner accès au secret identifiant le Client. Un exemple de Client s'exécutant dans un environnement sécurisé est une application s'exécutant sur une Web App localisée sur un serveur distant. Ceci n'est pas le cas lorsque le Client est une application Javascript ou une application native. Dans ces cas-ci il faut donc utiliser l'Implicit Grant Flow.

› **Client Credentials Grant Flow** : est nécessaire lorsque le Client est le Resource Owner.

Néanmoins, la méthode la plus sécurisée et donc celle à utiliser en priorité est celle de l'Authorization code.

Le flux approprié à une situation donnée peut être identifiée en se posant les questions suivantes :

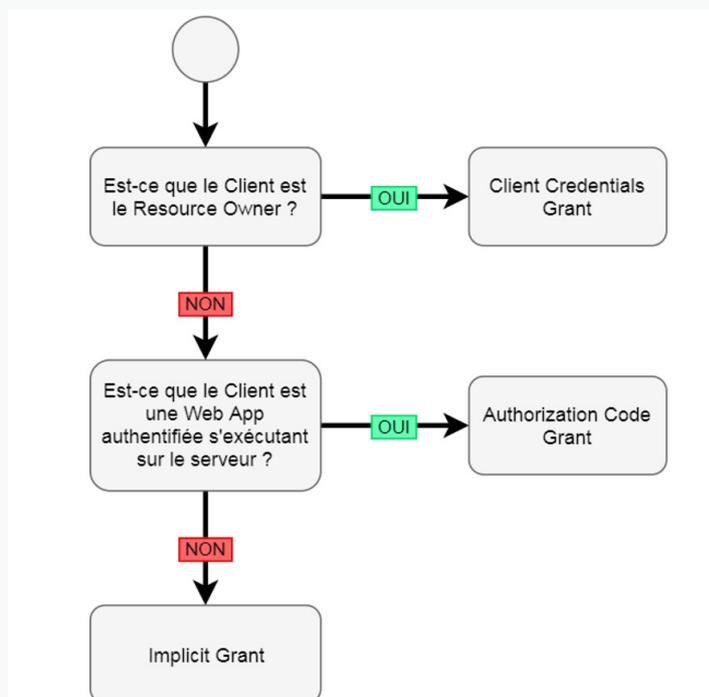


Figure 6 Diagramme de décision permettant de choisir le flux OAuth approprié

