GUIDE D'UTILISATION
AUTHENTIFICATION OAUTH

Version 2.0

Date d'entrée en vigueur : 08 juilllet 2016

SOMMAIRE

1 Introduction

Ce document décrit Les modalités d'authentification aux Api du Rte, via le protocole Oauth.

1.1 Documents de référence

Référence courte	Titre du document	Référence complète
[R1]	CGU des API RTE	Lien accès

1.2 Définitions

Les termes utilisés dans le Guide d'Utilisation et dont la première lettre est une majuscule sont définis ci-dessous ou, à défaut, dans les Conditions Générales d'Utilisation **[R1]** :

API	Application Programming Interface (Interface de programmation applicative)
Authentification	Mode de Protection permettant de s'assurer que l'identité de l'Émetteur ou du Récepteur a été vérifiée par RTE et qu'il est donc autorisé à accéder au SI et à utiliser les Applications.
URL	Uniform Resource Locator : chaîne de caractères suivant un format spécifique permettant de localiser une ressource sur un réseau et d'identifier un moyen d'agir (protocole) sur cette ressource.
Utilisateur(s)	Personne morale ayant validé les Conditions Générales d'Utilisation des API de RTE et accédant au SI de RTE afin d'utiliser les API mises à dispositions par RTE.
Web Service	Programme informatique permettant la communication entre systèmes hétérogènes dans des environnements distribués
OAuth 2	OAuth 2.0 (Open Authorization) est un framework de délégation d'autorisation qui permet à une application d'obtenir un jeton d'accès à des données hébergées par un tiers.

1.3 Assistance technique

En cas de difficulté pour l'accès ou l'utilisation d'une API, l'Utilisateur peut faire appel aux services d'assistance téléphonique mis en place par RTE dans les conditions techniques prévues dans les Conditions Générales d'Utilisation.

2 Accès aux API

2.1 Méthode d'Autorisation

Le mode d'autorisation mis en place s'appuie sur le framework de délégation d'autorisation OAuth 2.0 (Open Authorization), permettant à une application cliente d'accéder à une ressource exposée sous forme d'API au nom de son propriétaire, au travers d'un jeton d'accès, pour des données hébergées par un tier.

RFC OAuth 2.0: http://tools.ietf.org/html/rfc6749

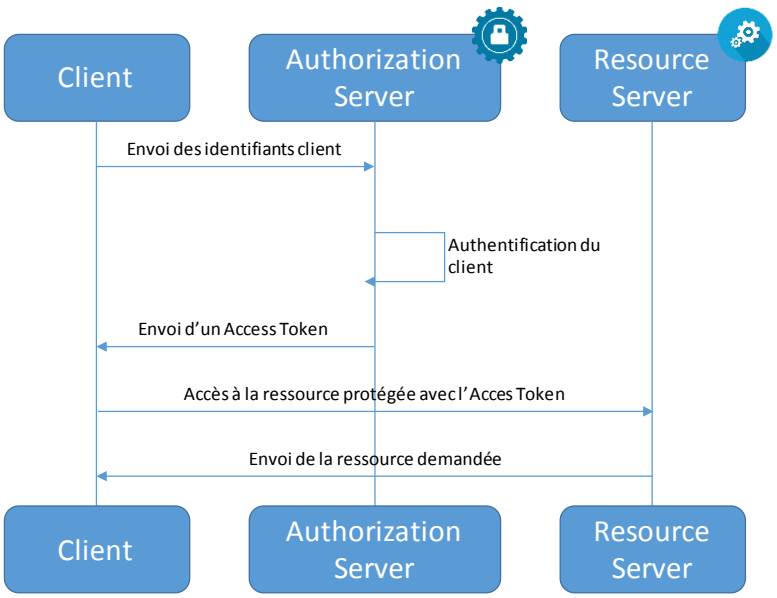
Dans les scénarios OAuth, on distingue 4 rôles:

- **Resource owner**: l'utilisateur final (Possesseur de la ressource)
- **Client**: Application ou Serveur demandant l'accès à une ressource protégée sur le Resource Server (Le client peut être une application java côté serveur, une application Javascript côté client ou une application native mobile par exemple).
- **Resource Server**: Serveur où se trouve la donnée protégée
- **Authorization Server**: Serveur délivrant les jetons. Ces tokens (access token) seront utilisés lors des requêtes du client vers le serveur de ressources

Oauth définit 4 scénarios possibles pour permettre au client d'obtenir un access token:

- Authorization Code
- Resource Owner Password Credentials
- Client Credentials
- Implicit Grant

Le scénario utilisé par les API du Rte est le Client Credentials :



2.1.1 Demande d'access token au serveur d'autorisation

La demande d'access token est faite par l'application cliente directement au serveur d'autorisation.

<table>
<tr><td>Endpoint</td><td colspan="4">digital.iservices.rte-france.com/token/oauth/</td></tr>
<tr><td>Méthode</td><td colspan="4">HTTP Method: POST
Content-Type: application/x-www-form-urlencoded</td></tr>
<tr><td rowspan="2">Paramètres</td><td>Nom</td><td>Type</td><td>Valeur</td><td>Cardinalité</td></tr>
<tr><td>Authorization</td><td>HEADER</td><td>«Basic»(1) suivi du couple client_id:client_secret encodé en base 64 (ces éléments, client_id et client_secret, sont fournis après abonnement de l'API à une application générée dans le portail d'API du Rte)</td><td>Requis</td></tr>
<tr><td>Exemple</td><td colspan="4">POST https://digital.iservices.rte-france.com/token/oauth/ HTTP/1.1
Authorization: Basic ZjdiZmExZWQtNmY0Py00YThkLTkxZjMtNGQ4NDczYmUwMTZjOjNhMkEyOTEyLWVlNTYtNDI1Zi1lMzZiLWI5NDY3OTFjMzQzYg==
Content-Type: application/x-www-form-urlencoded</td></tr>
</table>

Les attributs **grant_type** et **scope** ne sont pas à renseigner dans la requête d'invocation d'un token.

Le **scope** est géré automatiquement en fonction du profil utilisateur (utilisateur publique, utilisateur partenaire)

Le **grant_type** est déterminé par le **type d'application** (**Web/Serveur** ou **Mobile**) générée à partir du portail DATA et portant les informations de connexion (ClientID/ClientSecret), utilisés pour invoquer l'API.

(1) Attention, le contenu du header « Authorization » est sensible à la case, il faut donc respecter la case définie (exemple pour le « Basic », bien respecter le « B majuscule »).

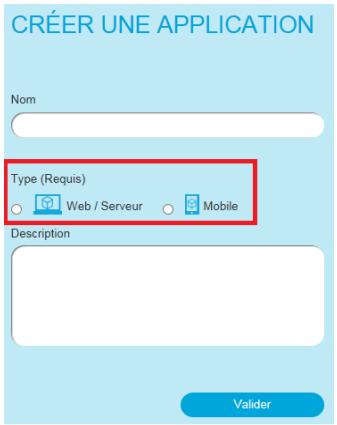
A la création d'une application sur le portail, il est demandé de choisir son type (**Web/Serveur** ou **Mobile**).
Chaque type d'application correspond à un flow Oauth2

Actuellement **seul le flow «client_credentials» est disponible**, via le type d'application « **Web/Serveur** ».
Le **type Mobile** est actuellement également associé au flow Oauth2 **«client_credentials»**, mais pourra être basculé par la suite sur un autre flow (« Resource Owner Password Credentials », par exemple).

Il est **ainsi recommandé de créer des applications de type « Web/Serveur »** lors du processus d'abonnement à une API.

Pour récupérer ses identifiants « **client_id** » et « **client_secret** », il faut tout d'abord s'abonner à l'API souhaitée en cliquant sur le bouton ci-dessous.

Puis créer une application en sélectionnant le type **« Web/Serveur ».** Dès qu'une première application est créée, celle-ci peut servir à accéder à différentes API.



Les identifiants sont ensuite accessibles via le menu « MES APPLICATIONS », en cliquant sur l'application précédemment créée.

En cas de problèmes de sécurité, les identifiants peuvent être réinitialisés.
Il est possible de créer plusieurs applications, qui auront chacune leurs identifiants propres.

2.1.2 Réponse d'access token du serveur d'autorisation

L'access token est renvoyé au format JSON dans le body de la réponse http.

<table>
<tr><td>Général</td><td colspan="3">Content-Type: application/json; charset=UTF-8</td></tr>
<tr><td rowspan="5">Paramètres</td><td>Nom</td><td>Contenu</td><td>Cardinalité</td></tr>
<tr><td>Access_token</td><td>L'access token.</td><td>Requis</td></tr>
<tr><td>Token_type</td><td>Ex : bearer</td><td>Requis</td></tr>
<tr><td>Expires_in</td><td>Nombre de secondes avant l'expiration de l'access token</td><td>Requis</td></tr>
<tr><td>scope</td><td>Liste de scope d'API pour lesquels l'access token est valide.</td><td>Optionnel</td></tr>
<tr><td>Exemple</td><td colspan="3">
<pre>
HTTP/1.1 200 OK
Date: Thu, 24 Dec 2015 14:26:59 GMT
X-CorrelationID: Id-b3007c56350554276222509d 0
Cache-Control: no-store
Content-Type: application/json
Pragma: no-cache
{
      "access_token":"kZBwyADEDgjYw4rADIWA0rPOtc9ULQ7FHdQZ
      2yWz9vxWseaihQU0IL",
      "token_type":"Bearer",
      "expires_in":7200,
}
</pre>
</td></tr>
</table>

2.1.3 Invocation de la ressource :

La demande d'accès à la ressource est faite par l'application cliente directement au serveur de ressource avec le token récupéré.

Dans cette partie, ne sont indiqué que les HEADER http nécessaire à l'appel OAUTH.

<table>
<tr><td>Endpoint</td><td colspan="4">digital.iservices.rte-france.com/[endpoint du service]</td></tr>
<tr><td>Méthode</td><td colspan="4">HTTP Method: POST
Content-Type: application/soap+xml;charset=UTF-8;</td></tr>
<tr><td rowspan="2">Paramètres</td><td>Nom</td><td>Type</td><td>Valeur</td><td>Cardinalité</td></tr>
<tr><td>Authorization</td><td>HEADER</td><td>« Bearer » suivi du token founi lors de l'appel au serveur d'autorisation</td><td>Requis</td></tr>
<tr><td>Exemple</td><td colspan="4">
<pre>
POST /privateapi/sandbox/getDonneesPhysiques/V1 HTTP/1.1
Content-Type: application/soap+xml;charset=UTF-8;
action="getDonneesPhysiquesAction"
Authorization: Bearer MUpX2RtZq5Gj1yAX1fhdsZ3qDr5dKhdVlfvpS
Host: 10.132.7.158:8066
Content-Length: 1103

<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
<soap:Body>
….
….
</soap:Body>
</soap:Envelope>
</pre>
</td></tr>
</table>

2.1.4 Erreurs possibles

Les erreurs rencontrées peuvent être les suivantes :

Codes d'erreur possibles	Description
invalid_request	Requête invalide pour cause de paramètre manquant, paramètre invalide, requête malformée, etc
invalid_client	L'authentification de l'application cliente a échoué
invalid_grant	Le grant_type est invalide, le code d'autorisation ou le refresh_token est invalide ou a expiré, ou la redirect_uri est invalide
unauthorized_client	L'application cliente n'est pas autorisée à demander un code d'autorisation
unsupported_grant_type	Le grant_type renseigné n'est pas supporté
invalid_token	Mauvais token ou token périmé fourni lors de l'appel à la ressource
invalid_scope	Le scope est invalide, inconnu ou malformé

Exemples (Pas de header http Authorization dans l'appel à la ressource)	`HTTP/1.1 401 Unauthorized`
	`Date: Tue, 06 Oct 2015 19:49:28 GMT`
	`Allow: OPTIONS, POST`
	`Server:`
	`Content-Length: 0`
	`Connection: close`
	`X-CorrelationID: Id-c8251456c9b142fd925aeb6b 0`
	`WWW-Authenticate: Bearer realm="DefaultRealm"`

Exemples (Dans le header http « Authorization » pas de « bearer » dans l'appel à la ressource)	`HTTP/1.1 400 Bad Request`
	`Date: Tue, 06 Oct 2015 19:52:59 GMT`
	`Allow: OPTIONS, POST`
	`Server:`
	`Content-Length: 0`
	`Connection: close`
	`X-CorrelationID: Id-9b26145635b274c6337385f7 0`
	`WWW-Authenticate: Bearer realm="DefaultRealm",error="invalid_request",error_description="Problem parsing the Authorization Header`

Exemples (Mauvais token ou token périmé fourni lors de l'appel à la ressource)	`HTTP/1.1 401 Unauthorized`
	`Date: Tue, 06 Oct 2015 19:51:11 GMT`
	`Allow: OPTIONS, POST`
	`Server:`
	`Content-Length: 0`
	`Connection: close`
	`X-CorrelationID: Id-2f261456feb15af315740ec1 0`
	`WWW-Authenticate: Bearer realm="DefaultRealm",error="invalid_token",error_description="Unable to find the access token in persistent storage."`

Exemples (Pas de définition du grant_type=client_credentials lors de l'appel à la ressource)	`HTTP/1.1 401 Unauthorized`
	`Date: Tue, 06 Oct 2015 20:10:16 GMT`
	`Server:`
	`Content-Encoding: gzip`
	`Connection: close`
	`X-CorrelationID: Id-a82a145637b3794468a87abd 0`

	Authorization: Basic ZWI4ODY3NTItNjJiZi00YjliLWExMjYtN2JhNTYyMzkyNjY3OjJmZDl jNjFhLWFjZWItNGEyYi1hODk5LTg2OTI2OWMwMmQzMg== Host: 10.132.7.157:8089 User-Agent: Jakarta Commons-HttpClient/3.1 Content-Type: text/html <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN"> <HTML><HEAD> <TITLE>401 Authorization Required</TITLE> </HEAD><BODY> <H1>Authorization Required</H1> This server could not verify that you are authorized to access the document requested. Either you supplied the wrong credentials (e.g., bad password), or your browser doesn't understand how to supply the credentials required.<P> <HR> </BODY></HTML>

Exemples
(Mauvais Authorization (ClientID:ClientSecret), Pas de basic ou pas de header Authorization défini lors de l'appel à la ressource)
	HTTP/1.1 401 Unauthorized
	Date: Tue, 06 Oct 2015 20:13:34 GMT
	Server:
	Content-Encoding: gzip
	Connection: close
	X-CorrelationID: Id-6e2b145669b322f541bb840c 0
	WWW-Authenticate: Basic realm="DefaultRealm"
	Content-Type: application/json
	{
	"error" : "invalid_client",
	"error_description" : "Client authentication failed (e.g. unknown client, no client authentication included, or unsupported authentication method). The authorization server MAY return an HTTP 401 (Unauthorized) status code to indicate which HTTP authentication schemes are supported. "
	}

~°~°~°~°~°~°~°~°~°~° Fin du document ~°~°~°~°~°~°~°~°~°~°