Utilisation de l'API Rest

pour utiliser, importer et exporter des données depuis et vers votre instance de Pod, vous avez deux possibilités : via un navigateur ou en ligne de commande.

1) Navigateur

Via votre navigateur, il vous suffit de vous rendre sur la page Rest de votre pod : http(s)://pod.univ.fr/rest et de renseigner le compte root de votre instance.

Vous aurez donc accès aux données au format JSON de votre instance et pourrez en poster de nouvelles.

N'hésitez pas à explorer cette interface pour y découvrir toutes les possibilités.

Nous vous conseillons de restreindre l'accès à "/rest" de votre instance via votre configuration nginx.

```
location /rest {
    allow XXX.XXX.X.X/24;
    deny all;
}
```

2) Terminal

Pour gérer les données de votre instance de Pod en ligne de commande, voici les différentes étapes à suivre :

- Dans l'administration, il faut créer un jeton d'authentification : http(s)://pod.univ.fr/admin/authtoken/

Attention, le jeton aura les mêmes accès que l'utilisateur sélectionné pour le créer.

Il vous suffit ensuite d'utiliser ce jeton dans vos requêtes Curl.

Par exemple, cette requête permet de récupérer les utilisateurs :

```
curl -H "Content-Type: application/json" -H 'Authorization: Token
XXXXXXXXXXX71922e47ed412eabcbd241XXXXXXX' -X GET -d '{}' http(s)://pod.univ.fr/rest/users/
```

Pour savoir comment créer vos requêtes, n'hésitez pas à utiliser l'interface web via votre navigateur, vous aurez la liste des objets modifiables et des exemples de requêtes.

Autre exemple, la commande suivante permet de créer un type intitulé "test :

```
curl -H "Content-Type: application/json" -H 'Authorization: Token XXXXXXXXXXX71922e47ed412eabcbd241XXXXXXX' -X POST -
d '{
   "title": "test"
}' http(s)://pod.univ.fr/rest/types/
```

L'exécution de cette commande renvoie le type créé :

```
{"id":13,"url":"http(s)://pod.univ.fr/rest/types/13/","title":"test","description":"-- désolé, aucune traduction fournie --","icon":null}
```

Enfin, on peut modifier un élément présent. Par exemple, on peut changer le type créé ci-dessus.

La commande suivante change le titre du type dont l'identifiant est 13

> *curl -H "Content-Type: application/json" -H 'Authorization: Token XXXXXXXXXXX71922e47ed412eabcbd241XXXXXXX' -X PATCH -d '{*
> *"title": "test new"*
> *}' http(s)://pod.univ.fr/rest/types/13/*

Cette commande renvoie les mêmes informations que lors de la création.

Enfin, il est également possible de poster (sans lancer l'encodage) des vidéos en ligne de commande. Voici un exemple :

> *curl -H "Content-Type: multipart/form-data" *
> *-H 'Authorization: Token XXXXXXXXXXX71922e47ed412eabcbd241XXXXXXX' *
> *-F "owner=http(s)://pod.univ.fr/rest/users/1/" *
> *-F "type=http(s)://pod.univ.fr/rest/types/1/" *
> *-F "title=ma video" *
> *-F "video=@/Users/test/video.mp4" *
> *http(s)://pod.univ.fr/rest/videos/*

En cas de succès, cette commande renvoie toutes les informations disponibles liées a cette vidéo. Si vous souhaitez lancer l'encodage de cette dernière, vous pouvez utiliser l'information "slug" ou titre-court (généré automatiquement lors de la création) en paramètre dans une deuxième commande. Exemple :

> *curl -XGET -H "Content-Type: application/json" *
> *-H 'Authorization: Token XXXXXXXXXXX71922e47ed412eabcbd241XXXXXXX' *
> *"http(s)://pod.univ.fr/rest/launch_encode_view/?slug=id-ma-video"*

Attention, pour les relations entre objet, il faut préciser l'url plutôt que la clé primaire :

> *The **HyperlinkedModelSerializer** class is similar to the **ModelSerializer** class except that it uses hyperlinks to represent relationships, rather than primary keys. By default the **serializer** will include a url **field** instead of a primary key **field**.*

Ceci pourrait être modifié dans les futures versions de Pod.

Gestion des diffuseurs en ligne de commande :

Récupération de la liste des diffuseurs :

```
curl  -H "Content-Type: application/json" \
  -H 'Authorization: Token XXXXXXXXXXX71922e47ed412eabcbd241XXXXXXX' \
  -F "slug=id-ma-video" \
  http(s)://pod.univ.fr/rest/broadcasters/
```

la réponse du serveur sera de cette forme :

```
{
"count": 1,
"next": null,
"previous": null,
"results": [
{
"id": 5,
"url": "https://pod.univ.fr/my_streamer/playlist.m3u8",
"name": "Nom de mon Diffuseur",
"slug": "nom-de-mon-diffuseur",
"building": "https://pod.univ.fr/rest/buildings/3/",
"description": "",
"poster": null,
"status": true
}
]
}
```

Vous trouverez ci-dessous un exemple de mise à jour du paramètre "status" d'un diffuseur. En utilisant la réponse précédente, on peut donc exécuter la commande suivante :

```
curl --location --request PATCH 'https://pod.univ.fr/rest/broadcasters/nom-de-mon-diffuseur/' \
--header 'Authorization: Token XXXXXXXXXXX71922e47ed412eabcbd241XXXXXXX' \
--header 'Content-Type: application/json' \
--data-raw '{"status" : true}'
```

3) DublinCore

Enfin, pour avoir la représentation au format DublinCore de vos vidéos, il suffit de faire une requete curl sur /rest/dublincore

Vous pouvez filtrer vos vidéos à l'aide de paramètre GET ajoutés à votre URL.

Par exemple, pour avoir la représentation DublinCore des vidéos de l'utilisateur 1, vous pouvez exécuter la commande suivante :

> *curl -H "Content-Type: application/json" -H 'Authorization: Token XXXXXXXXXXX71922e47ed412eabcbd241XXXXXXX' -X GET http(s)://pod.univ.fr/rest/dublincore/?owner=1*

Ceci vous renverra un XML au format DublinCore.

Attention a bien renseigner les variables DEFAULT_DC_COVERAGE et DEFAULT_DC_RIGHTS dans votre fichier de configuration.