Aide-mémoire – API de CKAN

Le présent document résume les éléments d'intérêt pour les utilisateurs de Données Québec, en vue de pouvoir utiliser les API de CKAN.

Table des matières

1 Qu'est-ce qu'une API?

L'expression « API » est l'abréviation de « Application Programming Interface » (ou en français: interface applicative de programmation). Les API sont un moyen efficace de faire communiquer deux programmes informatiques[1].

2 Quelle est son utilité?

Une API permet d'automatiser certaines tâches dynamiques en lot et de programmer des tâches qui doivent être exécutées fréquemment.

Par exemple, un programme informatique pourrait utiliser une API de CKAN afin d'extraire la liste des établissements condamnés pour les manquements à la réglementation sur les produits alimentaires et utiliser ces données pour géolocaliser les restaurants.

3 Description des API de CKAN

Le portail Données Québec est soutenu par l'application Web CKAN, qui permet de cataloguer et de documenter les jeux de données, ainsi que de téléverser leurs fichiers de données d'un serveur distant vers le serveur de Données Québec. CKAN possède des API qui permettent d'interagir par programmation avec les données du portail.

3.1 Utilisateurs et mode d'accès

Deux types d'utilisateurs auront recours aux API de Données Québec : les consommateurs de données (citoyens, développeurs, etc.) et les diffuseurs (ministères, organismes publics, municipalités et organismes de la société civile). En fonction de leurs besoins et de leurs droits, deux modes d'accès peuvent leur être attribués :

1. en **lecture** pour les consommateurs et diffuseurs;
2. en **écriture** pour les diffuseurs. Ce mode requiert une clé API.

3.1.1 Mode lecture

En mode lecture, les API de CKAN permettent de **consulter** le contenu du portail: le catalogue des jeux de données, les organisations, l'historique, les ressources associées, etc. Par exemple, un développeur pourrait extraire la liste des jeux de données présents dans le portail au moyen de l'API générique. Évidemment, les actions en lecture seulement ne permettent en aucun cas de modifier le contenu des jeux de données.
Note : un jeu de données est un ensemble de ressources sur un sujet et une ressource est un fichier.

[1] Inspiré de https://www.1min30.com/dictionnaire-du-web/api-interface-de-programmation.

3.1.2 Mode écriture

En mode écriture, les API de CKAN permettent aux organisations d'apporter des changements (**ajouter, modifier, supprimer**) au contenu du portail, entre autres :

- créer un nouveau jeu de données
- mettre à jour les jeux de données d'une organisation à fréquence déterminée;
- téléverser la mise à jour d'un fichier de données de manière ponctuelle;
- mettre à jour l'information de la fiche de métadonnées du jeu de données;
- supprimer une ressource.

En général, les API permettent de poser les mêmes actions que celles pouvant être réalisées par l'interface utilisateur (à l'aide des formulaires). Toutes les spécifications techniques liées au portail Données Québec (ex. taille maximale de fichiers, timeout, etc.) s'appliquent autant lors du recours aux API qu'à l'interface utilisateur.

En mode écriture, une clé API est toujours requise pour authentifier l'utilisateur. Certaines API en mode lecture requièrent aussi une clé API. Une clé API unique est associée à chaque compte utilisateur dans CKAN. Pour l'obtenir, l'utilisateur doit se connecter au portail et accéder à la page de son compte (la clé API se situe au bas à gauche de cette page) :
https://www.donneesquebec.ca/recherche/user/**votreNomUtilisateur.**

Les actions en écriture sont possibles par un utilisateur sur un jeu de données lorsque les droits lui sont accordés par l'administrateur de l'organisation propriétaire du jeu de données.

3.2 Catégories principales d'API de CKAN

CKAN propose trois catégories d'API pour interagir avec la majorité de ses composants : API générique, API DatasStore, API FileStore.

3.2.1 API génériques

Les API génériques permettent d'interroger, de modifier, d'ajouter ou de supprimer des composants de tout portail de données CKAN, y inclus : **les organisations, les catégories, les mots-clés, licences, jeux de données, les métadonnées associées, etc.**
La documentation complète de CKAN est disponible sur le Web (en anglais)**.**

Les API génériques sont regroupées en fonction du type d'action exécutée sur le composant de CKAN. Nous avons cinq groupes d'actions : GET, CREATE, UPDATE, PATCH, et DELETE.

Les actions **GET**, permettent de faire des recherches et des extractions de métadonnées.
Exemples :

1. **package_list** fournit la liste des jeux de données du portail;
2. **package_show** permet d'obtenir les métadonnées d'un jeu de données;
3. **resource_show** fournit les métadonnées sur une ressource;
4. **package_search** fournit la liste des jeux de données satisfaisant les critères de recherche avec leurs métadonnées;
5. **current_package_list_with_resources** retourne la liste des jeux de données du portail avec leurs ressources;
6. **group_list** donne la liste des catégories du portail;
7. **license_list** pour avoir la liste des licences sous lesquelles les données peuvent être publiées.

Pour des détails sur les API GET, consulter la documentation CKAN (en anglais).

Les actions **CREATE** permettent d'ajouter des nouveaux éléments ou données.
Exemples :

1. **package_create** pour l'ajout d'un nouveau jeu de données;
2. **resource_create** pour l'ajout d'une nouvelle ressource;
3. **user_create** pour l'ajout d'un nouvel utilisateur;
4. **member_create** pour l'ajout d'un nouveau membre (utilisateur, jeu de données, groupe) à un groupe.

Pour des détails sur les API CREATE, consulter la documentation CKAN (en anglais).

Les actions **UPDATE** permettent de mettre à jour l'information existante.
Exemples :

1. **package_update** pour mettre à jour les métadonnées d'un jeu de données;
2. **resource_update** pour modifier les données et métadonnées d'une ressource;
3. **package_resource_reorder** pour réorganiser l'ordre d'affichage des ressources d'un jeu de données;
4. **user_generate_apikey** pour change la clé API d'un utilisateur.

Pour des détails sur les API UPDATE, consulter la documentation CKAN (en anglais)

Les actions **PATCH** permettent de mettre à jour partiellement l'information existante.
Exemples :

1. **package_patch** pour modifier uniquement les champs passés en paramètre dans un jeu de données;
2. **resource_patch** pour modifier uniquement les champs passés en paramètre dans une ressource;
3. **organization_patch** pour modifier uniquement les champs passés en paramètre dans une organisation.

Pour des détails sur les API PATCH, consulter la documentation CKAN (en anglais).

Note : Il est important de comprendre la différence entre les actions UPDATE et PATCH. La méthode PATCH fait une mise à jour à partir des paramètres fournis, en laissant les autres paramètres inchangés, tandis que la méthode UPDATE efface les paramètres qui ne sont pas explicitement fournis. Pour les mises à jour de ressources ou une modification des métadonnées, il est recommandé d'utiliser l'action PATCH.

Les actions **DELETE** permettent de supprimer des objets de CKAN.
Exemples :

1. **resource_delete** pour supprimer une ressource;
2. **package_delete** pour changer le statut un jeu de données à invisible et le rendre inaccessible par Web et API;
3. **organisation_member_delete** pour retirer un utilisateur d'une organisation.

Pour des détails sur les API PATCH, consulter la documentation CKAN (en anglais).

Il est important de souligner que dans les API génériques, les jeux de données sont appelés "package" et non "dataset". Les actions des API génériques sont détaillées dans la documentation CKAN (en anglais).

3.2.2 API FileStore

Le FileStore est un espace disque distant ou local où les ressources sont stockées. Pour en savoir plus, consultez la documentation CKAN (en anglais).

Les API FileStore permettent d'ajouter et de mettre à jour tous les types de **fichiers téléversés** dans CKAN. Les ressources du FileStore sont accessibles en intégralité et ne peuvent être interrogées pour en extraire une partie. Cette possibilité est explorée dans la section suivante, API Datastore.

Que le fichier soit téléversé par l'API, ou manuellement par l'interface utilisateur, la bonne pratique consiste à conserver le même nom de fichier lors de la mise à jour, afin de permettre aux applications développées avec ces fichiers d'aller chercher la mise à jour des données à partir du même URL de manière automatisée.

Deux actions sont permises pour téléverser un fichier dans le FileStore: "**resource_create**" et "**resource_update**". Pour en savoir plus sur les actions de l'API FileStore, vous pouvez consulter la documentation CKAN (en anglais).

De plus, la version téléchargeable d'un fichier (peu importe son format) ne sera pas automatiquement importée dans l'infrastructure géomatique ouverte (IGO) lorsque ce fichier est utilisé comme source d'une carte interactive. Il faut donc importer le fichier IGO indépendamment de l'API.

3.2.3 API DataStore

Le DataStore est une base de données de CKAN utilisée pour le stockage des ressources ayant des données structurées. Les ressources sont stockées sous forme de tables. Le but est de faciliter l'accès aux données en permettant de faire des manipulations (lecture, écriture, filtrage, ajout, modification, suppression) sur les ressources disponibles dans le Datastore sans avoir à les télécharger.

Les ressources au format CSV, XLS, ou XLSX téléversées dans CKAN (FileStore) sont automatiquement importées dans le DataStore, pour pouvoir être exploitées. Il faut cependant noter que les actions apportées dans la base de données DataStore ne mettent pas à jour les versions téléchargeables (les fichiers dans le FileStore).

Voici quelques actions qu'on peut exécuter sur une ressource du DataStore par API :

1. **datastore_search** pour rechercher des données dans une ressource;
2. **datastore_search_sql** pour interroger une ou plusieurs ressources avec des requêtes SQL;
3. **datastore_upsert** pour ajouter ou mettre à jour des données dans une ressource;
4. **datastore_delete** pour supprimer une ressource du DataStore.

Pour plus de détails sur les actions des API DataStore, consulter la documentation CKAN (en anglais).

4 Environnements de travail de Données Québec

Les organisations diffusant sur Données Québec peuvent effectuer des tests avec l'interface graphique ou les requêtes d'API sur l'environnement de développement. Cet environnement est accessible pour des adresses IP spécifiées. Il faut donc fournir l'adresse IP de l'organisation, se créer un compte et demander les accès en édition pour celle-ci.

Rappelons que pour le site public, seules les requêtes d'API en lecture sont permises pour tous les utilisateurs. Pour les organisations diffusant sur Données Québec, un accès en écriture doit être demandé pour l'organisation diffusant les jeux de données afin obtenir une clé API.

Les demandes d'accès doivent être acheminées à pilote@donneesquebec.ca.

URL des environnements pour l'utilisation des API

Environnement de développement: http://beta.donneesquebec.ca/recherche/api/3/action/**NOM_API**
Environnement public : https://www.donneesquebec.ca/recherche/api/3/action/**NOM_API**

5 Exemples de requêtes API

Les interactions avec l'API se font par requêtes HTTP au moyen d'instructions qui utilisent les méthodes:

- GET: lorsque l'API se limite à un accès en lecture seulement;
- POST: lorsque l'API nécessite un accès en écriture (et requiert la clé API).

L'exécution des requêtes API fournit en sorties des fichiers au format JSON, avec la structure suivante :

```
{  "help" : " "
   "success" : true ou false
      "error" ou "result" :{

          //result contient le resutat de la requête pour "success":true

          //error contient le message d'erreur et le type d'erreur pour
          //"success":false

      }
 }
```

Il existe plusieurs outils permettant l'exécution des requêtes API. Les plus utilisés pour les requêtes CKAN sont : Navigateur (Chrome, Internet Explorer, Fire Fox), Curl, Httpie, Postman.

5.1 API génériques

1. Lister toutes les catégories du portail Données Québec (méthode GET) avec **group_list** :

https://www.donneesquebec.ca/recherche/api/action/group_list
curl https://www.donneesquebec.ca/recherche/api/action/group_list


```
{
  help: "https://www.donneesquebec.ca/recherche/api/3/action/help_show?name=group_list",
  success: true,
  result:

      [

        "agriculture-alimentation",
        "economie-entreprises",
        "education-recherche",
        "environnement-ressources-naturelles-energie",
        "gouvernement-finances",
        "infrastructures",
        "loi-justice-securite-publique",
        "politiques-sociales",
```

```
        "sante",
        "societe-culture",
        "tourisme-sports-loisirs",
        "transport"

      ]

}
```


2. Lister tous les jeux de données (appelés « package » avec l'API) de Données Québec (méthode GET) avec **package_list** :

https://www.donneesquebec.ca/recherche/api/action/package_list
curl https://www.donneesquebec.ca/recherche/api/action/package_list

3. Faire une recherche des jeux de données contenant le mot « pompier » (méthode GET) avec **package_search** :

https://www.donneesquebec.ca/recherche/api/3/action/package_search?q=pompier
curl https://www.donneesquebec.ca/recherche/api/3/action/package_search?q=pompier

4. Afficher les métadonnées du jeu de données nommé « arbres » (le nom peut être remplacé par l'identifiant du jeu : 9ed153b2-4751-4e03-862f-6d4027e6f2a6) (méthode GET) avec **package_show** :

https://www.donneesquebec.ca/recherche/api/3/action/package_show?id=arbres
https://www.donneesquebec.ca/recherche/api/3/action/package_show?id=9ed153b2-4751-4e03-862f-6d4027e6f2a6
curl https://www.donneesquebec.ca/recherche/api/3/action/package_show?id=arbres


```
{
    help: "https://www.donneesquebec.ca/recherche/api/3/action/help_show?name=package_show",
    success: true,
    result:
    {
            license_title: "Creative Commons 4.0 – Attribution CC BY",
            maintainer: null,
            extras_organisation_principale: "ville-de-longueuil",
```

relationships_as_object: [ ],
private: false,
maintainer_email: null,
num_tags: 4,
update_frequency: "semiannual",
id: "9ed153b2-4751-4e03-862f-6d4027e6f2a6",
metadata_created: "2016-06-30T12:40:50.063386",
metadata_modified: "2018-07-20T12:32:52.611723",
author: "Service de la géomatique, Direction de l'aménagement et de l'urbanisme",
author_email: "donneesouvertes@longueuil.quebec",
temporal: "",
state: "active",
version: null,
license_id: "cc-by",
type: "dataset",
resources:
[
+ { ...},
+ {...},
+ {...}
],
num_resources: 3,
tags:
[
+ {...},
+ {...},
+ {...},
+ {...}
],
language: "FR",
methodologie: "",
groups:
[
{
display_name: "Environnement, ressources naturelles et énergie",
description: "Domaine d'affaires correspondant à la mise en valeur, l'utilisation optimale du territoire et des ressources énergétiques, forestières et minérales et ce, dans une perspective de développement durable. Cela inclut forêts, gisements de pétrole, de gaz naturel ou de minerai, ressources hydro-électriques et autres biens de même nature qui ont une valeur économique certaine.",
image_display_url: "https://www.donneesquebec.ca/recherche/images/categories/environnement-ressources-naturelles-energie.gif",
title: "Environnement, ressources naturelles et énergie",
id: "d61e0cae-e722-4000-8fb1-88b28399e20c",
name: "environnement-ressources-naturelles-energie"
}
],
creator_user_id: "6dae464f-b8e9-4024-b26a-e58c62617945",
relationships_as_subject: [ ],
ext_spatial: "ville-longueuil",
organization:

```
            {
                "description": "La Ville de Longueuil s'est engagée dans l'ouverture de données publiques. Elle prône ainsi une gestion municipale ouverte et transparente et une modernisation des relations avec les citoyens. Longueuil invite les curieux et les développeurs à télécharger et à s'approprier les jeux issus de ses bases de données. Cette initiative visant à favoriser la transparence, l'innovation technologique ainsi que la création de nouveaux services pour les citoyens.",
                "created": "2016-09-19T14:26:33.357677",
                "title": "Ville de Longueuil",
                "name": "ville-de-longueuil",
                "is_organization": true,
                "state": "active",
                "image_url": "2016-09-19-183314.655491signaturelongueuil.png",
                "revision_id": "9a94d808-8687-43c0-bc91-e26fd87f16c4",
                "type": "organization",
                "id": "856b172f-18c3-4a5b-a2bc-ecdccae651d3",
                "approval_status": "approved"
            },
            "name": "arbres",
            "isopen": true,
            "url": "",
            "notes": "L'ensemble des arbres, principalement municipaux, inventoriés à ce jour. L'inventaire est en cours de réalisation, il n'est donc pas nécessairement complet.",
            "owner_org": "856b172f-18c3-4a5b-a2bc-ecdccae651d3",
            "license_url": "https://www.donneesquebec.ca/fr/licence/#cc-by",
            "title": "Arbres",
            "revision_id": "d2f0f498-98c6-4795-98b9-6b04fdd72f14"
        }
}
```


L'action **package_show** donne des informations sur les métadonnées du jeu. Ces informations seront utiles lors de la mise à jour du jeu.

5. Création d'un nouveau jeu de données (méthode POST) avec **package_create** :

```
curl -X POST -H Authorization:CleAPI https://beta.donneesquebec.ca/recherche/api/3/action/package_create -F name=jeu-cree-par-api -F title="Jeu de données crée par API 4" -F notes="Description du jeu de données" -F license_id=cc-by -F update_frequency=monthly -F private=False -F owner_org=2c8496be-f549-4c92-8be3-867c948435c0 -F extras_organisation_principale=gouvernement-du-quebec -F author="Secrétariat du Conseil du Trésor, DRGOLL" -F author_email=pilote@donnesquecbec.ca -F ext_spatial=province-quebec -F language=FR_EN
```

6. Mise à jour du jeu dont l'identifiant est 14d93d62-16bf-48da-ae4f-4be21a32e2ad (méthode POST) avec **package_update** (paramètres similaires à ceux de package_create) :

```
curl -X POST -H Authorization:CleAPI https://beta.donneesquebec.ca/recherche/api/3/action/package_update -F id=14d93d62-16bf-48da-ae4f-4be21a32e2ad -F update_frequency=quarterly -F notes= "__Jeu de données à haute valeur__ ## Critique sur les usages des ressources minière" -F maintainer=DRGOLL -F license_id=cc-by -F ext_spatial=province-quebec -F name=registre-des-entreprises-non-admissibles-aux-contrats-publics-rena -F owner_org=2c8496be-f549-4c92-8be3-867c948435c0 -F title="Registre des entreprises non admissibles aux contrats publics (RENA)" -F language=FR -F extras_organisation_principale=gouvernement-du-quebec -F author="Secrétariat du Conseil du trésor, Sous-secrétariat aux marchés publics" -F author_email=rena@sct.gouv.qc.ca
```

Il faut que toutes les valeurs obligatoires soient passées en paramètre sinon la mise à jour ne s'effectuera pas. Les paramètres non indiqués prendront les valeurs par défaut si définies. Pour changer un ou plusieurs paramètres des métadonnées d'un jeu de données sans modifier d'autres valeurs, il est préférable d'utiliser l'API **package_patch.**

7. Modification de la fréquence de mise à jour du jeu dont l'identifiant est 14d93d62-16bf-48da-ae4f-4be21a32e2ad (méthode POST) **package_patch** :

```
curl -X POST -H Authorization:CleAPI https://beta.donneesquebec.ca/recherche/api/3/action/package_patch -F id=14d93d62-16bf-48da-ae4f-4be21a32e2ad -F update_frequency=monthly
```

8. Suppression du jeu de données (méthode POST) avec **package_delete** :

```
curl -X POST -H Authorization:CleAPI https://beta.donneesquebec.ca/recherche/api/3/action/package_delete -F id=6569e0d5-0326-4ba0-8472-2e0465b7097a
```

9. Ajout de la catégorie **santé** au jeu de données dont l'identifiant est e193a73d-8569-4a7d-98fb-6e0e4bf33938 (méthode POST) **member_create** :

```
curl -X POST \ https://beta.donneesquebec.ca/recherche/api/3/action/member_create -H authorization: CleAPI -F id=sante –F object=6569e0d5-0326-4ba0-8472-2e0465b7097a -F object_type=package -F capacity=member
```

10. Afficher les métadonnées de la ressource dont l'identifiant est 23cde69a-a1d7-4775-8271-e3b46b3a6d83 (méthode GET) avec **resource_show**.

https://www.donneesquebec.ca/recherche/api/3/action/resource_show?id=23cde69a-a1d7-4775-8271-e3b46b3a6d83
curl https://www.donneesquebec.ca/recherche/api/3/action/resource_show?id=23cde69a-a1d7-4775-8271-e3b46b3a6d83


```
{"help": "https://www.donneesquebec.ca/recherche/api/3/action/help_show?name=resource_show",
"success": true,
"result": {
        "cache_last_updated": null,
        "package_id": "9ed153b2-4751-4e03-862f-6d4027e6f2a6",
        "webstore_last_updated": null,
        "datastore_active": false,
        "id": "23cde69a-a1d7-4775-8271-e3b46b3a6d83",
        "description": "",
        "state": "active",
        "size": null,
        "hash": "",
        "standardqc": "",
        "format": "JSON",
        "last_modified": null,
        "url_type": null,
        "mimetype": null,
        "cache_url": null,
        "name": "Arbres",
        "created": "2016-06-30T08:41:58.867216",
        "url": "https://www.longueuil.quebec/sites/longueuil/files/donnees_ouvertes/arbres.json",
        "webstore_url": null,
        "mimetype_inner": null,
        "taille_entier": 7,
        "position": 0,
        "revision_id": "d2f0f498-98c6-4795-98b9-6b04fdd72f14",
        "resource_type": "donnees"
        }
}
```


11. Pour l'action **resource_create**
    https://www.donneesquébec.ca/recherche/api/3/action/resource-create (Voir API FileStore)

12. Mise à jour du fichier associé à la ressource et à la taille du fichier (méthode POST) avec **resource_patch** :

```
curl -X POST -H Authorization:CleAPI
https://www.donneesquebec.ca/recherche/api/3/action/resource_patch  -F id=8d0f521b-50d5-444d-a552-cf7b2d996e6a  -F url=upload  -F taille_entier=6 -F upload=@cheminAu Fichier\NouveauFichier.csv
```

13. Mise à jour de la date de dernière modification la ressource (méthode POST) avec **resource_patch**. Cette méthode pourra être appliquée par les diffuseurs ayant les données hébergées à distance:

```
curl -X POST -H Authorization:CleAPI
https://www.donneesquebec.ca/recherche/api/3/action/resource_patch  -F id=8d0f521b-50d5-444d-a552-cf7b2d996e6a  -F last_modified =2019-05-30T12:38:00
```

Note: Par souci de cohérence avec les autres dates du portail, il est important d'indiquer le temps en UTC.

14. Suppression du jeu d'une ressource (méthode POST) avec **resource_delete** :

```
curl -X POST -H Authorization:CleAPI
https://beta.donneesquebec.ca/recherche/api/3/action/resource_delete -F id=8d0f521b-50d5-444d-a552-cf7b2d996e6a
```

5.2 API DataStore

1. Faire une recherche sur les cinq premiers éléments de la ressource « Liste mensuelle des 250 termes les plus recherchés sur le portail de BAnQ» dont l'identifiant est 0c8f6384-63b1-4520-8c05-25af7b69c856 (méthode GET) avec **datastore_search** :

https://www.donneesquebec.ca/recherche/api/action/datastore_search?resource_id=0c8f6384-63b1-4520-8c05-25af7b69c856&limit=5

curl https://www.donneesquebec.ca/recherche/api/action/datastore_search?resource_id=0c8f6384-63b1-4520-8c05-25af7b69c856&limit=5

2. Lister les termes dont le nombre total de recherches uniques est supérieur à 50 (méthode GET) avec **datastore_search_sql** :

https://www.donneesquebec.ca/recherche/api/action/datastore_search_sql?sql=SELECT%20*%20FROM%20%220c8f6384-63b1-4520-8c05-25af7b69c856%22%20WHERE%20%22Nombre_total_de_recherches_uniques%22%3E50

```
curl
https://www.donneesquebec.ca/recherche/api/action/datastore_search_sql?sql=SELECT%20*%20FROM%20%220c8f6384-63b1-4520-8c05-25af7b69c856%22%20WHERE%20%22Nombre_total_de_recherches_uniques%22%3E50
```

3. Supprimer la ressource dont l'identifiant est 19385b4e-5503-4330-9e59-f998f5918363 du DataStore (méthode POST) avec **datastore_delete** :

```
https://beta.donneesquebec.ca/recherche/api/action/datastore_delete?resource_id=19385b4e-5503-4330-9e59-f998f5918363

curl -X POST -H Authorization:CleAPI https://beta.donneesquebec.ca/recherche/api/action/datastore_delete?resource_id=19385b4e-5503-4330-9e59-f998f5918363 - F force=true
```

5.3 API FileStore

1. Création d'une nouvelle ressource pour le jeu de données dont l'identifiant est avec la **resource_create** :

```
curl -X POST -H Authorization:CleAPI https://beta.donneesquebec.ca/recherche/api/3/action/resource_create -F package_id=14d93d62-16bf-48da-ae4f-4be21a32e2ad -F name=ressource_cree_api -F url=upload -F description="Fichier CSV contenant la liste des jeux de données " -F taille_entier=6 -F resource_type=donnees -F relidi_condon_valinc=oui -F upload=@C:\chemin\fichier.csv
```

2. Téléverser un nouveau fichier associé à une ressource pour un jeu de données existant, avec la ligne de commande (méthode POST) **resource_update** :

```
curl -X POST -H Authorization:CleAPI https://beta.donneesquebec.ca/recherche/api/3/action/resource_pupdate -F id=8d0f521b-50d5-444d-a552-cf7b2d996e6a -F name=ressource_cree_api -F url=upload -F description="Fichier CSV contenant la liste des jeux de données " -F taille_entier=6 -F resource_type=donnees -F relidi_condon_valinc=oui -F upload=@C:\chemin\NewFile.csv
```

6 Encodage

L'encodage utilisé par défaut par les API est UTF-8. L'utilisation d'autres encodages (ex.: ANSI, ISO-8859-1) peut engendrer des problèmes de compatibilité. Par exemple, les caractères latins « é » et « à » risquent d'être convertis en caractères non conformes ("Ã" et "?").

Pour pallier ce problème, il faut s'assurer:

a. de spécifier l'encodage UTF-8 dans l'instruction de requête à l'API si les fichiers de données ne sont pas déjà encodés en UTF-8;
b. que le fichier au format JSON contenant les instructions de requête à l'API soit également encodé en UTF-8, sinon, les caractères accentués ne seront pas correctement interprétés, puisque convertis en caractères non conformes.

7 Soutien technique

Pour toute demande d'assistance sur les API, veuillez envoyer un courriel au pilote de Données Québec (pilote@donneesquebec.ca).

ANNEXE

Tableau 1 : Liste des champs spécifiques au portail de Données Québec pour les jeux de données

Nom champs interface graphique	**Nom du paramètre pour l'API**	**Valeurs possibles**	**Commentaire**
Couverture géographique	**ext_spatial**	province-quebec	Territoire couvert par les données
		ville-blainville	
		ville-gatineau	
		ville-laval	
		ville-longueuil	
		ville-montreal	
		ville-quebec	
		ville-repentigny	
		ville-rimouski	
		ville-rouyn-noranda	
		ville-saint-felicien	
		ville-shawinigan	
		ville-sherbrooke	
		autre	
Langue	**language**	FR	Langue française
		FR_EN	Langue française et anglaise
Organisation Principale	**extras_organisation_principale**	gouvernement-du-quebec	Représente l'organisation abritant une ou plusieurs autres organisations appelées « organisation participante »
		organismes	
		ville-de-blainville	
		ville-de-gatineau	
		ville-de-laval	
		ville-de-longueuil	
		ville-de-montreal	
		ville-de-quebec	
		ville-de-repentigny	
		ville-de-rimouski	
		ville-de-rouyn-noranda	
		ville-de-shawinigan	

		ville-de-sherbrooke	
Couverture temporelle	**temporal**	Date	
Information complémentaire	**methodologie**	Texte	Texte donnant des informations complémentaires sur le jeu de données. Le texte peut être formaté en utilisant les Markdowns.

Tableau 2 : Liste des champs spécifiques au portail de Données Québec pour les ressources.

Nom champs interface graphique	**Nom du paramètre pour l'API**	**Valeurs possibles**	**Commentaire**
Taille du fichier	**taille_entier**	valeur entière	Indique la taille du fichier. Cette information est importante pour le téléchargement.
Type de ressource	**resource_type**	donnees	La ressource contient de la donnée
		support	Documentation ou guide
		web	Service Web
		cartes	Carte interactive

Tableau 3 : Liste des champs spécifiques au portail de Données Québec pour respecter des lignes directrices au niveau des ressources

Nom champs interface graphique	**Nom du paramètre pour l'API**	**Valeurs possibles**	**Commentaire**
Description des champs	**relidi_description_champs**	relidi.descha.foudoc	Description des champs fournie dans la ressource
		relidi.descha.foumet	Description des champs fournie dans la métadonnée
		relidi.descha.absent	Pas de description de champs
Ressource complémentaire	**relidi_ressource_complementaire**	[relidi.rescom]	Pour indiquer que la ressource est complémentaire. La liste vide dans le cas contraire.
Valeur inconnue / non disponible = vide	**relidi_condon_valinc**	oui	Pas de valeurs inconnues ou valeurs inconnues remplacer par le vide
		non	Fichier contient des valeurs inconnues indiquées par des valeurs autres que le vide
		n/a	Non applicable
Nombre (décimal, unité SI et monétaire)	**relidi_condon_nombre**	oui	Utilisation du point pour unité décimale, des mesures du SI ou du Dollar Canadien (CAD) comme monnaie
		non	Non-respect des Normes sur les nombres
		n/a	Non applicable
Booléen	**relidi_condon_boolee**	oui	Les booléens sont représentés par des valeurs telles que (TRUE, FALSE), (Vrai, Faux) ou (oui, non)

		non	Les booléens sont représentés par des valeurs autres que celle ci-dessus
		n/a	Non applicable
Date et heure	**relidi_condon_datheu**	oui	Les dates et heures suivent l'un des formats suivants : (AAAA-MM-JJ/HH:MM:SS AAAA-MM-JJTHH:MM:SS)
		non	Les formats ci-dessus ne sont pas appliqués sur les dates et heures
		n/a	Non applicable
UTF8/Balise UTF8	**relidi_confic_utf8**	oui	Encodage UTF8 utilisé
		non	Pas d'encodage UTF8
		n/a	Non applicable
Séparateur de virgule	**relidi_confic_separateur_virgule**	oui	La virgule utilisée comme séparateur de champs
		non	Autre caractère utilisé colonne
		n/a	Non applicable
Pas de compression sauf SHP, GTFS, GBFS	**relidi_confic_pascom**	oui	Pas de compression autre que pour les fichiers SHP, GTFS et GBFS
		non	Compression appliquée sur les fichiers autres que ceux indiqués ci-dessus.
		n/a	Non applicable
ESPG : 4326/ESPG : 32198	**Relidi_confic_epsg**	oui	Système de coordonnées géographiques ESPG 43265 ou ESPG 32198
		non	Système de coordonnées géographiques autre que ci-dessus
		n/a	Non applicable