.. \_action-api:

L’API d’action

==============

L’API d’action CKAN est une puissante interface d’application de type RPC qui met à la disposition des clients API les caractéristiques fondamentales de CKAN. Toutes les fonctionnalités fondamentales du site Web CKAN (c’est-à-dire tout ce que vous pouvez réaliser par l’intermédiaire de l’interface Web, et plus encore) peuvent être exécutées à l’aide d’un programme externe qui fait appel à l’API CKAN. Par des exemples, de ce que peut réaliser votre application à l’aide de cette API.

\* Obtenir une liste au format JSON des jeux de données, des groupes ou d’autres objets JSON d’un site Web :

http://demo.ckan.org/api/3/action/package\_list

http://demo.ckan.org/api/3/action/group\_list

http://demo.ckan.org/api/3/action/tag\_list

\* Obtenir une représentation JSON exhaustive d’un jeu de données, d’une ressource ou d’un autre objet :

http://demo.ckan.org/api/3/action/package\_show?id=adur\_district\_spending

http://demo.ckan.org/api/3/action/tag\_show?id=gold

http://demo.ckan.org/api/3/action/group\_show?id=data-explorer

\* Rechercher des ensembles ou des ressources qui correspondent à une requête donnée :

http://demo.ckan.org/api/3/action/package\_search?q=dépenses

http://demo.ckan.org/api/3/action/resource\_search?query=name:noms%20de%20district

\* Créer, actualiser et supprimer des jeux de données, des groupes ou d’autres objets.

\* Obtenir une lecture en continu des modifications récemment apportées aux jeux de données d’un site Web :

http://demo.ckan.org/api/3/action/recently\_changed\_packages\_activity\_list

Pour faire une requête API

---------------------

Pour faire appel à l’API CKAN, vous devez transmettre un dictionnaire JSON dans une requête HTTP de type POST à l’une des adresses URL de l’API CKAN. Ce dictionnaire doit inclure les paramètres requis par la fonction invoquée. La réponse CKAN est elle aussi retournée dans un dictionnaire JSON.

Il est possible d’envoyer un dictionnaire JSON à une adresse URL à l’aide de la ligne de commande du client HTTP en utilisant l’instruction `HTTPie <http://httpie.org/>`\_. Par exemple, pour obtenir la liste de tous les noms des jeux de données du groupe ``data-explorer``, sur le site demo.ckan.org, installez HTTPie sur un terminal et appelez la fonction ``group\_list`` de l’API en exécutant la commande ci-dessous.

http http://demo.ckan.org/api/3/action/group\_list id=data-explorer

La réponse CKAN sera similaire à celle illustrée ci-dessous.

{

"help": "...",

"result": [

"data-explorer",

"department-of-ricky",

"geo-examples",

"geothermal-data",

"reykjavik",

"skeenawild-conservation-trust"

],

"success": true

}

La réponse est un dictionnaire JSON constitué de trios clés.

1. ``"sucess"``: ``true`` ou ``false``.

L’API retourne toujours l’expression ``200 OK`` comme code d’état de sa réponse HTTP, peu importe qu’il y ait ou non des erreurs dans la requête. C’est pourquoi il est important de vérifier la valeur de la clé ``"success"`` dans le dictionnaire retourné par la fonction et (si cette valeur est ``False``) de vérifier la valeur le la clé ``"error"``.

.. note::

Si un problème majeur de formatage survient dans une requête transmise à l’API, CKAN peut renvoyer une réponse HTTP dont le code d’état peut être ``409``, ``400`` ou ``500`` (par ordre croissant de gravité). Dans les versions ultérieures de CKAN nous entendons inhiber ce type de réponses afin de toujours renvoyer le code d’état ``200 OK`` et d’utiliser les clés

``"success"`` et ``"error"``.

2. ``"result"`` : exprime le résultat retourné par la fonction invoquée. Le type et la valeur de cette clé varient selon la fonction. Dans le cas de la fonction ``group\_list``, il s’agit d’une liste de chaînes de caractères représentant les noms de tous les jeux de données appartenant au groupe visé.

Si une erreur est survenue dans la réponse à votre requête, au lieu de la clé ``"result"``,le dictionnaire contiendra une clé ``"error"`` qui donne les détails concernant l’erreur en question. Voici une illustration d’une réponse qui contient une clé d’erreur.

{

"help": "Creates a package",

"success": false,

"error": {

"message": "Access denied",

"\_\_type": "Authorization Error"

}

}

3. ``"help"`` : chaîne de caractères repésentant la documentation relative à la function invoquée.

On peut effectuer la meme requête HTTP à l’aide du module standard ``urllib2`` de Python en se servant des instructions ci-après.

#!/usr/bin/env python

import urllib2

import urllib

import json

import pprint

# Utiliser le module JSON module pour convertir un dictionnaire

# en une chaîne de caractères transmissible dans une requête POST.

data\_string = urllib.quote(json.dumps({'id': 'data-explorer'}))

# Lancer la requête HTTP.

response = urllib2.urlopen('http://demo.ckan.org/api/3/action/group\_list',

data\_string)

assert response.code == 200

# Utiliser le module JSON module pour importer la réponse CKAN dans un dictionnaire.

response\_dict = json.loads(response.read())

# Vérifier le contenu de la réponse.

assert response\_dict['success'] is True

result = response\_dict['result']

pprint.pprint(result)

Version de l’API

------------

Les API CKAN sont identifiées par un numéro de version. Si vous transmettez une requête à l’adresse URL d’une API sans préciser son numéro de version, CKAN utilisera la plus récente version de cette API.

http://demo.ckan.org/api/action/package\_list

Vous avez toutefois la possibilité de préciser le numéro de version voulu dans l’adresse URL de votre requête.

http://demo.ckan.org/api/3/action/package\_list

Pour l’instant, seule la version 3 de l’API d’action est accessible.

Nous recommandons d’indiquer le numéro de version de l’API dans les requêtes parce que l’on s’assure ainsi que le client API sera compatible avec différents sites qui exécutent différentes versions de CKAN (et qu’il continuera d’être compatible advenant une mise à niveau de ces sites vers des versions ultérieures). Étant donné que la plus récente version de l’API peut changer lorsqu’un site migre vers une nouvelle version ou que la version peut différer d’un site à l’autre selon la version de CKAN en cours d’exécution, les résultats d’une requête qui n’indique pas le numéro de version de l’API ne sont pas nécessairement fiables.

Authentication et clés de l’API

---------------------------

Certaines fonctions de l’API exigent une permission d’accès. L’API a recours aux mêmes fonctions d’autorisation et de configuration que celles qu’utilise l’interface Web, de sorte que si vous êtes un utilisateur autorisé à effectuer certaines actions dans l’interface Web, vous avez aussi la permission de les effectuer par l’intermédiaire de l’API.

Lorsque vous invoquez une fonction de l’API qui exige une permission d’accès, vous devez vous authentifier en incluant votre clé d’API dans votre requête HTTP. Pour connaître votre clé d’API, ouvrez une session sur le site Web de CKAN, par le biais de l’interface Web, et consultez la page de votre profil utilisateur

Pour ajouter votre clé d’API dans une requête HTTP, incluez-la dans un en-tête ``Authorization`` ou ``X-CKAN-API-Key``. (Le nom de l’en-tête HTTP peut être configuré à l’aide de l’option ``apikey\_header\_name``, dans votre fichier de configuration CKAN.)

Par exemple, pour demander si vous suivez ou non l’utilisateur ``markw`` sur le site demo.ckan.org à l’aide de HTTPie, lancez cette commande :

http http://demo.ckan.org/api/3/action/am\_following\_user id=markw Authorization:XXX

(en y remplaçant l’expression ``XXX`` par votre proper clé d’API.)

Ou encore, pour obtenir la liste des activités en cours sur votre tableau de bord d’utilisateur du site demo.ckan.org, exécutez les instructions Python ci-dessous.

request = urllib2.Request('http://demo.ckan.org/api/3/action/dashboard\_activity\_list')

request.add\_header('Authorization', 'XXX')

response\_dict = json.loads(urllib2.urlopen(request, '{}').read())

Fonctions API accessibles par la méthode GET

-------------------------------------------

Les fonctions définies dans le document :doc:`ckan.logic.action.get` peuvent aussi être invoquées à l’aide d’une requête HTTP utilisant la méthode GET. Par exemple, pour obtenir la liste des jeux de données (packages) du site demo.ckan.org, ouvrez cette adresse URL dans votre navigateur Web :

http://demo.ckan.org/api/3/action/package\_list

Ou encore, pour rechercher tous les jeux de données (packages) détecté par à une recherche sur le mot ``spending`` dans le site demo.ckan.org, ouvrez cette adresse URL dans votre navigateur Web :

http://demo.ckan.org/api/3/action/package\_search?q=spending

.. tip::

Des plugiciels comme `JSONView for Firefox <https://addons.mozilla.org/en-us/firefox/addon/jsonview/>`\_ ou `Chrome <https://chrome.google.com/webstore/detail/jsonview/chklaanhfefbnpoihckbnefhakgolnmc>`\_ permettent aux navigateurs Web de mettre en forme et de surligner en couleurs de façon adéquate les réponses renvoyées par le module JSON de CKAN.

L’argument de recherche est indiqué dans l’adresse URL par le paramètre ``?q=dépenses``. On peut ajouter d’autres paramètres en les délimitant par le caractère ``&``. Par exemple, pour retenir uniquement les dix premiers jeux de données correspondants, ouvrez cette adresse URL :

http://demo.ckan.org/api/3/action/package\_search?q=dépenses&rows=10

Si une action exige une valeur de paramètre composée d’une liste de chaînes de caractères, on peut la transmettant en répétant plusieurs fois le nom de ce paramètre dans l’adresse URL.

http://demo.ckan.org/api/3/action/term\_translation\_show?terms=russie&terms=roman%20d’amour

Prise en charge de JSONP

------------------------

Pour tenir compte des scripts provenant d’autres sites Web qui accèdent à l’API, les données peuvent être retournées au format JSONP, dans lequel les données JSON sont imbriquées dans l’appel de fonction. Cette fonction est désignée par le paramètre 'callback'. Voici un exemple.

http://demo.ckan.org/api/3/action/package\_show?id=adur\_district\_spending&callback=myfunction

.. todo :: This doesn't work with all functions.

.. \_api-reference:

Documentation de l’API d’action

--------------------

.. This hidden toctree is just to shut up sphinx warnings about the following

files not being included in any toctree. We want to include them manually

because we're using a different style to what the toctree would use.

.. toctree::

:maxdepth: 1

:hidden:

ckan.logic.action.get

ckan.logic.action.create

ckan.logic.action.update

ckan.logic.action.delete

Functions for getting data from CKAN: :doc:`ckan.logic.action.get`.

Functions for adding data to CKAN: :doc:`ckan.logic.action.create`.

Functions for updating existing data in CKAN: :doc:`ckan.logic.action.update`.

Functions for deleting data from CKAN: :doc:`ckan.logic.action.delete`.

Functions for working with the CKAN DataStore: :doc:`datastore-api`.