.. \_action-api:

The Action API

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

CKAN's Action API is a powerful, RPC-style API that exposes all of CKAN's core

features to API clients. All of a CKAN website's core functionality (everything

you can do with the web interface and more) can be used by external code that

calls the CKAN API. For example, using the CKAN API your app can:

\* Get JSON-formatted lists of a site's datasets, groups or other CKAN objects:

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

\* Get a full JSON representation of a dataset, resource or other object:

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

\* Search for packages or resources matching a query:

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

http://demo.ckan.org/api/3/action/resource\_search?query=name:District%20Names

\* Create, update and delete datasets, resources and other objects

\* Get an activity stream of recently changed datasets on a site:

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

Making an API Request

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

To call the CKAN API, post a JSON dictionary in an HTTP POST request to one of

CKAN's API URLs. The parameters for the API function should be given in the

JSON dictionary. CKAN will also return its response in a JSON dictionary.

One way to post a JSON dictionary to a URL is using the command-line HTTP

client `HTTPie <http://httpie.org/>`\_. For example, to get a list of the names

of all the datasets in the ``data-explorer`` group on demo.ckan.org, install

HTTPie and then call the ``group\_list`` API function by running this command

in a terminal::

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

The response from CKAN will look like this::

{

"help": "...",

"result": [

"data-explorer",

"department-of-ricky",

"geo-examples",

"geothermal-data",

"reykjavik",

"skeenawild-conservation-trust"

],

"success": true

}

The response is a JSON dictionary with three keys:

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

The API aims to always return ``200 OK`` as the status code of its HTTP

response, whether there were errors with the request or not, so it's

important to always check the value of the ``"success"`` key in the response

dictionary and (if success is ``False``) check the value of the ``"error"``

key.

.. note::

If there are major formatting problems with a request to the API, CKAN

may still return an HTTP response with a ``409``, ``400`` or ``500``

status code (in increasing order of severity). In future CKAN versions

we intend to remove these responses, and instead send a ``200 OK``

response and use the ``"success"`` and ``"error"`` items.

2. ``"result"``: the returned result from the function you called. The type

and value of the result depend on which function you called. In the case of

the ``group\_list`` function it's a list of strings, the names of all the

datasets that belong to the group.

If there was an error responding to your request, the dictionary will

contain an ``"error"`` key with details of the error instead of the

``"result"`` key. A response dictionary containing an error will look like

this::

{

"help": "Creates a package",

"success": false,

"error": {

"message": "Access denied",

"\_\_type": "Authorization Error"

}

}

3. ``"help"``: the documentation string for the function you called.

The same HTTP request can be made using Python's standard ``urllib2`` module,

with this Python code::

#!/usr/bin/env python

import urllib2

import urllib

import json

import pprint

# Use the json module to dump a dictionary to a string for posting.

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

# Make the HTTP request.

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

data\_string)

assert response.code == 200

# Use the json module to load CKAN's response into a dictionary.

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

# Check the contents of the response.

assert response\_dict['success'] is True

result = response\_dict['result']

pprint.pprint(result)

API Versions

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

The CKAN APIs are versioned. If you make a request to an API URL without a

version number, CKAN will choose the latest version of the API::

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

Alternatively, you can specify the desired API version number in the URL that

you request::

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

Version 3 is currently the only version of the Action API.

We recommend that you specify the API number in your requests, because this

ensures that your API client will work across different sites running different

version of CKAN (and will keep working on the same sites, when those sites

upgrade to new versions of CKAN). Because the latest version of the API may

change when a site is upgraded to a new version of CKAN, or may differ on

different sites running different versions of CKAN, the result of an API

request that doesn't specify the API version number cannot be relied on.

Authentication and API Keys

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

Some API functions require authorization. The API uses the same authorization

functions and configuration as the web interface, so if a user is authorized to

do something in the web interface they'll be authorized to do it via the API as

well.

When calling an API function that requires authorization, you must authenticate

yourself by providing your API key with your HTTP request. To find your API

key, login to the CKAN site using its web interface and visit your user profile

page.

To provide your API key in an HTTP request, include it in either an

``Authorization`` or ``X-CKAN-API-Key`` header. (The name of the HTTP header

can be configured with the ``apikey\_header\_name`` option in your CKAN

configuration file.)

For example, to ask whether or not you're currently following the user

``markw`` on demo.ckan.org using HTTPie, run this command::

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

(Replacing ``XXX`` with your API key.)

Or, to get the list of activities from your user dashboard on demo.ckan.org,

run this Python code::

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())

GET-able API Functions

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

Functions defined in :doc:`ckan.logic.action.get` can also be called with an HTTP

GET request. For example, to get the list of datasets (packages) from

demo.ckan.org, open this URL in your browser:

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

Or, to search for datasets (packages) matching the search query ``spending``,

on demo.ckan.org, open this URL in your browser:

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

.. tip::

Browser plugins like `JSONView for Firefox <https://addons.mozilla.org/en-us/firefox/addon/jsonview/>`\_

or `Chrome <https://chrome.google.com/webstore/detail/jsonview/chklaanhfefbnpoihckbnefhakgolnmc>`\_

will format and color CKAN's JSON response nicely in your browser.

The search query is given as a URL parameter ``?q=spending``. Multiple

URL parameters can be appended, separated by ``&`` characters, for example

to get only the first 10 matching datasets open this URL:

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

When an action requires a list of strings as the value of a parameter, the

value can be sent by giving the parameter multiple times in the URL:

http://demo.ckan.org/api/3/action/term\_translation\_show?terms=russian&terms=romantic%20novel

JSONP Support

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

To cater for scripts from other sites that wish to access the API, the data can

be returned in JSONP format, where the JSON data is 'padded' with a function

call. The function is named in the 'callback' parameter. For example:

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:

Action API Reference

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

.. 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`.