CHAPTER 4

API guide

This section documents CKAN's API, for developers who want to write code that interacts with CKAN sites and their data.

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

Note: CKAN's FileStore and DataStore have their own APIs, see:

- *FileStore and file uploads*
- *DataStore extension*

Note: For documentation of CKAN's legacy API's, see *Legacy APIs*.

4.1 Legacy APIs

Warning: The legacy APIs documented in this section are provided for backwards-compatibility, but support for new CKAN features will not be added to these APIs. These endpoints will be removed in the future.

Note: The REST API was deprecated in CKAN v2.0 and removed starting from CKAN v2.8.

4.1.1 Search API

Search resources are available at published locations. They are represented with a variety of data formats. Each resource location supports a number of methods.

The data formats of the requests and the responses are defined below.

Search Resources

Here are the published resources of the Search API.

Search Resource	Location
Dataset Search	`/search/dataset`
Resource Search	`/search/resource`
Revision Search	`/search/revision`
Tag Counts	`/tag_counts`

See below for more information about dataset and revision search parameters.

Search Methods

Here are the methods of the Search API.

Resource	Method	Request	Response
Dataset Search	POST	Dataset-Search-Params	Dataset-Search-Response
Resource Search	POST	Resource-Search-Params	Resource-Search-Response
Revision Search	POST	Revision-Search-Params	Revision-List
Tag Counts	GET		Tag-Count-List

It is also possible to supply the search parameters in the URL of a GET request, for example `/api/search/dataset?q=geodata&allfields=1`.

Search Formats

Here are the data formats for the Search API.

Name	Format
Dataset-Search-Params Resource-Search-Params Revision-Search-Params	{ Param-Key: Param-Value, Param-Key: Param-Value, ... } See below for full details of search parameters across the various domain objects.
Dataset-Search-Response	{ count: Count-int, results: [Dataset, Dataset, ...] }
Resource-Search-Response	{ count: Count-int, results: [Resource, Resource, ...] }
Revision-List	[Revision-Id, Revision-Id, Revision-Id, ...] NB: Ordered with youngest revision first. NB: Limited to 50 results at a time.
Tag-Count-List	[[Name-String, Integer], [Name-String, Integer], ...]

Dataset Parameters

Param-Key	Param-Value	Examples	Notes
q	Search-String	q=geodata q=government+sweden q=%22drug%20abuse%22 q=tags:"river pollution"	Criteria to search the dataset fields for. URL-encoded search text. (You can also concatenate words with a '+' symbol in a URL.) Search results must contain all the specified words. You can also search within specific fields.
qjson	JSON encoded options	['q':'geodata']	All search parameters can be json-encoded and supplied to this parameter as a more flexible alternative in GET requests.
title, tags, notes, groups, author, maintainer, update_frequency, or any 'extra' field name e.g. department	Search-String	title=uk&tags=health department=environment tags=health&tags=pollution tags=river%20pollution	Search in a particular a field.
order_by	field-name (default=rank)	order_by=name	Specify either rank or the field to sort the results by
offset, limit	result-int (defaults: offset=0, limit=20)	offset=40&limit=20	Pagination options. Offset is the number of the first result and limit is the number of results to return.
all_fields	0 (default) or 1	all_fields=1	Each matching search result is given as either a dataset name (0) or the full dataset record (1).

Note: filter_by_openness and filter_by_downloadable were dropped from CKAN version 1.5 onwards.

Note: Only public datasets can be accessed via the legacy search API, regardless of the provided authorization. If you need to access private datasets via the API you will need to use the *package_search* method of the *API guide*.

Resource Parameters

Param-Key	Param-Value	Example	Notes
url, format, description	Search-String	url=statistics.org format=xls description=Research+Institute	Criteria to search the dataset fields for. URL-encoded search text. This search string must be found somewhere within the field to match. Case insensitive.
qjson	JSON encoded options	[‘url’:’www.statistics.org’]	All search parameters can be json-encoded and supplied to this parameter as a more flexible alternative in GET requests.
hash	Search-String	hash=b0d7c260-35d4-42ab-9e3d-c1f4db9bc2f0	Searches for an match of the hash field. An exact match or match up to the length of the hash given.
all_fields	0 (default) or 1	all_fields=1	Each matching search result is given as either an ID (0) or the full resource record
offset, limit	result-int (defaults: offset=0, limit=20)	offset=40&limit=20	Pagination options. Offset is the number of the first result and limit is the number of results to return.

Note: Powerful searching from the command-line can be achieved with curl and the qjson parameter. In this case you need to remember to escapt the curly braces and use url encoding (e.g. spaces become `%20`). For example:

```
curl 'http://thedatahub.org/api/search/dataset?qjson=\{"author":"The%20Stationery
↪%20Office%20Limited"\}'
```

Revision Parameters

Param-Key	Param-Value	Example	Notes
since_time	Date-Time	since_time=2010-05-05T19:42:45.854533	The time can be less precisely stated (e.g 2010-05-05).
since_id	Uuid	since_id=6c9f32ef-1f93-4b2f-891b-fd01924ebe08	The stated id will not be included in the results.

4.1.2 Util API

The Util API provides various utility APIs – e.g. auto-completion APIs used by front-end javascript.

All Util APIs are read-only. The response format is JSON. Javascript calls may want to use the JSONP formatting.

dataset autocomplete

There an autocomplete API for package names which matches on name or title.

This URL:

```
/api/2/util/dataset/autocomplete?incomplete=a%20novel
```

Returns:

```
{"ResultSet": {"Result": [{"match_field": "title", "match_displayed": "A Novel By␣
↪Tolstoy (annakarenina)", "name": "annakarenina", "title": "A Novel By Tolstoy"}]}}
```

tag autocomplete

There is also an autocomplete API for tags which looks like this:

This URL:

```
/api/2/util/tag/autocomplete?incomplete=ru
```

Returns:

```
{"ResultSet": {"Result": [{"Name": "russian"}]}}
```

resource format autocomplete

Similarly, there is an autocomplete API for the resource format field which is available at:

```
/api/2/util/resource/format_autocomplete?incomplete=cs
```

This returns:

```
{"ResultSet": {"Result": [{"Format": "csv"}]}}
```

munge package name

For taking an readable identifier and munging it to ensure it is a valid dataset id. Symbols and whitespeace are converted into dashes. Example:

```
/api/util/dataset/munge_name?name=police%20spending%20figures%202009
```

Returns:

```
"police-spending-figures-2009"
```

munge title to package name

For taking a title of a package and munging it to a readable and valid dataset id. Symbols and whitespeace are converted into dashes, with multiple dashes collapsed. Ensures that long titles with a year at the end preserves the year should it need to be shortened. Example:

```
/api/util/dataset/munge_title_to_name?title=police:%20spending%20figures%202009
```

Returns:

```
"police-spending-figures-2009"
```

munge tag

For taking a readable word/phrase and munging it to a valid tag (name). Symbols and whitespeace are converted into dashes. Example:

```
/api/util/tag/munge?tag=water%20quality
```

Returns:

```
"water-quality"
```

4.1.3 Status Codes

Standard HTTP status codes are used to signal method outcomes.

Code	Name
200	OK
201	OK and new object created (referred to in the Location header)
301	Moved Permanently
400	Bad Request
403	Not Authorized
404	Not Found
409	Conflict (e.g. name already exists)
500	Service Error

Note: On early CKAN versions, datasets were called "packages" and this name has stuck in some places, specially internally and on API calls. Package has exactly the same meaning as "dataset".

4.2 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 client Curl. For example, to get a list of the names of all the datasets in the `data-explorer` group on demo.ckan.org, install curl and then call the `group_list` API function by running this command in a terminal:

```
curl https://demo.ckan.org/api/3/action/group_list
```

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. `"success"`: `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

# Make the HTTP request.
response = urllib2.urlopen('http://demo.ckan.org/api/3/action/group_list',
```

(continues on next page)

(continued from previous page)

```
        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)
```

4.3 Example: Importing datasets with the CKAN API

You can add datasets using CKAN's web interface, but when importing many datasets it's usually more efficient to automate the process in some way. In this example, we'll show you how to use the CKAN API to write a Python script to import datasets into CKAN.

Todo: Make this script more interesting (eg. read data from a CSV file), and all put the script in a .py file somewhere with tests and import it here.

```
#!/usr/bin/env python
import urllib2
import urllib
import json
import pprint

# Put the details of the dataset we're going to create into a dict.
dataset_dict = {
    'name': 'my_dataset_name',
    'notes': 'A long description of my dataset',
    'owner_org': 'org_id_or_name'
}

# Use the json module to dump the dictionary to a string for posting.
data_string = urllib.quote(json.dumps(dataset_dict))

# We'll use the package_create function to create a new dataset.
request = urllib2.Request(
    'http://www.my_ckan_site.com/api/action/package_create')

# Creating a dataset requires an authorization header.
# Replace *** with your API key, from your user account on the CKAN site
# that you're creating the dataset on.
request.add_header('Authorization', '***')

# Make the HTTP request.
response = urllib2.urlopen(request, data_string)
assert response.code == 200

# Use the json module to load CKAN's response into a dictionary.
response_dict = json.loads(response.read())
assert response_dict['success'] is True
```

(continues on next page)

(continued from previous page)

```
# package_create returns the created package as its result.
created_package = response_dict['result']
pprint.pprint(created_package)
```

For more examples, see *API Examples*.

4.4 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.

4.5 Authentication and API tokens

Warning: Starting from CKAN 2.9, API tokens are the preferred way of authenticating API calls. The old legacy API keys will still work but they will be removed in future versions so it is recommended to switch to use API tokens. Read below for more details.

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 an authentication key with your HTTP request. Starting from CKAN 2.9 the recommended mechanism to use are API tokens. These are encrypted keys that can be generated manually from the UI (User Profile > Manage > API tokens) or via the `api_token_create()` function. A user can create as many tokens as needed for different uses, and revoke one or multiple tokens at any time. In addition, enabling the `expire_api_token` core plugin allows to define the expiration timestamp for a token.

Site maintainers can use *API Token Settings* to configure the token generation.

Legacy API keys (UUIDs that look like *ec5c0860-9e48-41f3-8850-4a7128b18df8*) are still supported, but its use is discouraged as they are not as secure as tokens and are limited to one per user. Support for legacy API keys will be removed in future CKAN versions.

To provide your API token 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 curl, run this command:

```
curl -H "Authorization: XXX"  https://demo.ckan.org/api/3/action/am_following_user?
↪id=markw
```

(Replacing `XXX` with your API token.)

Or, to get the list of activities from your user dashboard on demo.ckan.org, run this Python code:

```
request = urllib2.Request('https://demo.ckan.org/api/3/action/dashboard_activity_list
↪')
request.add_header('Authorization', 'XXX')
response_dict = json.loads(urllib2.urlopen(request, '{}').read())
```

4.6 GET-able API functions

Functions defined in *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 or Chrome 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

4.7 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

---

Note: This only works for GET requests

4.8 API Examples

4.8.1 Tags (not in a vocabulary)

A list of all tags:

- browser: http://demo.ckan.org/api/3/action/tag_list
- curl: `curl http://demo.ckan.org/api/3/action/tag_list`
- ckanapi: `ckanapi -r http://demo.ckan.org action tag_list`

Top 10 tags used by datasets:

- browser: http://demo.ckan.org/api/action/package_search?facet.field={[}%22tags%22{]}&facet.limit=10&rows=0
- curl: `curl 'http://demo.ckan.org/api/action/package_search?facet.field=\["tags"\]&facet.limit=10&rows=0'`
- ckanapi: `ckanapi -r http://demo.ckan.org action package_search facet.field='["tags"]' facet.limit=10 rows=0`

All datasets that have tag 'economy':

- browser: http://demo.ckan.org/api/3/action/package_search?fq=tags:economy
- curl: `curl 'http://demo.ckan.org/api/3/action/package_search?fq=tags:economy'`
- ckanapi: `ckanapi -r http://demo.ckan.org action package_search fq='tags:economy'`

4.8.2 Tag Vocabularies

Top 10 tags and vocabulary tags used by datasets:

- browser: http://demo.ckan.org/api/action/package_search?facet.field={[}%22tags%22{]}&facet.limit=10&rows=0
- curl: `curl 'http://demo.ckan.org/api/action/package_search?facet.field=\["tags"\]&facet.limit=10&rows=0'`
- ckanapi: `ckanapi -r http://demo.ckan.org action package_search facet.field='["tags"]' facet.limit=10 rows=0`

e.g. Facet: *vocab_Topics* means there is a vocabulary called Topics, and its top tags are listed under it.

A list of datasets using tag 'education' from vocabulary 'Topics':

- browser: https://data.hdx.rwlabs.org/api/3/action/package_search?fq=vocab_Topics:education
- curl: `curl 'https://data.hdx.rwlabs.org/api/3/action/package_search?fq=vocab_Topics:education'`
- ckanapi: `ckanapi -r https://data.hdx.rwlabs.org action package_search fq='vocab_Topics:education'`

4.8.3 Uploading a new version of a resource file

You can use the `upload` parameter of the *`resource_patch()`* function to upload a new version of a resource file. This requires a `multipart/form-data` request, with curl you can do this using the `@file.csv`:

```
curl -X POST  -H "Content-Type: multipart/form-data"  -H "Authorization: XXXX"  -F
↪"id=<resource_id>" -F "upload=@updated_file.csv" https://demo.ckan.org/api/3/action/
↪resource_patch
```

4.9 Action API reference

Note: If you call one of the action functions listed below and the function raises an exception, the API will return a JSON dictionary with keys `"success": false` and an `"error"` key indicating the exception that was raised.

For example *`member_list()`* (which returns a list of the members of a group) raises `NotFound` if the group doesn't exist. If you called it over the API, you'd get back a JSON dict like this:

```
{
    "success": false
    "error": {
        "__type": "Not Found Error",
        "message": "Not found"
    },
    "help": "...",
}
```

4.9.1 ckan.logic.action.get

API functions for searching for and getting data from CKAN.

`ckan.logic.action.get.`**`site_read`**(*context*, *data_dict=None*)
Return `True`.

Return type bool

`ckan.logic.action.get.`**`package_list`**(*context*, *data_dict*)
Return a list of the names of the site's datasets (packages).

Parameters

- **`limit`** (*int*) – if given, the list of datasets will be broken into pages of at most `limit` datasets per page and only one page will be returned at a time (optional)
- **`offset`** (*int*) – when `limit` is given, the offset to start returning packages from

Return type list of strings

`ckan.logic.action.get.`**`current_package_list_with_resources`**(*context*, *data_dict*)
Return a list of the site's datasets (packages) and their resources.

The list is sorted most-recently-modified first.

Parameters

- **`limit`** (*int*) – if given, the list of datasets will be broken into pages of at most `limit` datasets per page and only one page will be returned at a time (optional)

- **offset** (*int*) – when `limit` is given, the offset to start returning packages from
- **page** (*int*) – when `limit` is given, which page to return, Deprecated: use `offset`

Return type list of dictionaries

`ckan.logic.action.get.`**`member_list`**(*context*, *data_dict=None*)
Return the members of a group.

The user must have permission to 'get' the group.

Parameters

- **id** (*string*) – the id or name of the group
- **object_type** (*string*) – restrict the members returned to those of a given type, e.g. `'user'` or `'package'` (optional, default: `None`)
- **capacity** (*string*) – restrict the members returned to those with a given capacity, e.g. `'member'`, `'editor'`, `'admin'`, `'public'`, `'private'` (optional, default: `None`)

Return type list of (id, type, capacity) tuples

Raises `ckan.logic.NotFound`: if the group doesn't exist

`ckan.logic.action.get.`**`package_collaborator_list`**(*context*, *data_dict*)
Return the list of all collaborators for a given package.

Currently you must be an Admin on the package owner organization to manage collaborators.

Note: This action requires the collaborators feature to be enabled with the *ckan.auth.allow_dataset_collaborators* configuration option.

Parameters

- **id** (*string*) – the id or name of the package
- **capacity** (*string*) – (optional) If provided, only users with this capacity are returned

Returns a list of collaborators, each a dict including the package and user id, the capacity and the last modified date

Return type list of dictionaries

`ckan.logic.action.get.`**`package_collaborator_list_for_user`**(*context*, *data_dict*)
Return a list of all package the user is a collaborator in

Note: This action requires the collaborators feature to be enabled with the *ckan.auth.allow_dataset_collaborators* configuration option.

Parameters

- **id** (*string*) – the id or name of the user
- **capacity** (*string*) – (optional) If provided, only packages where the user has this capacity are returned

Returns a list of packages, each a dict including the package id, the capacity and the last modified date

Return type list of dictionaries

`ckan.logic.action.get.`**`group_list`**(*context*, *data_dict*)
Return a list of the names of the site's groups.

Parameters

- **order_by** (*string*) – the field to sort the list by, must be `'name'` or `'packages'` (optional, default: `'name'`) Deprecated use sort.
- **sort** (*string*) – sorting of the search results. Optional. Default: "title asc" string of field name and sort-order. The allowed fields are 'name', 'package_count' and 'title'
- **limit** (*int*) – the maximum number of groups returned (optional) Default: `1000` when all_fields=false unless set in site's configuration `ckan.group_and_organization_list_max` Default: `25` when all_fields=true unless set in site's configuration `ckan.group_and_organization_list_all_fields_max`
- **offset** (*int*) – when `limit` is given, the offset to start returning groups from
- **groups** (*list of strings*) – a list of names of the groups to return, if given only groups whose names are in this list will be returned (optional)
- **all_fields** (*bool*) – return group dictionaries instead of just names. Only core fields are returned - get some more using the include_* options. Returning a list of packages is too expensive, so the *packages* property for each group is deprecated, but there is a count of the packages in the *package_count* property. (optional, default: `False`)
- **include_dataset_count** (*bool*) – if all_fields, include the full package_count (optional, default: `True`)
- **include_extras** (*bool*) – if all_fields, include the group extra fields (optional, default: `False`)
- **include_tags** (*bool*) – if all_fields, include the group tags (optional, default: `False`)
- **include_groups** (*bool*) – if all_fields, include the groups the groups are in (optional, default: `False`).
- **include_users** (*bool*) – if all_fields, include the group users (optional, default: `False`).

Return type list of strings

`ckan.logic.action.get.`**organization_list**(*context*, *data_dict*)

Return a list of the names of the site's organizations.

Parameters

- **order_by** (*string*) – the field to sort the list by, must be `'name'` or `'packages'` (optional, default: `'name'`) Deprecated use sort.
- **sort** (*string*) – sorting of the search results. Optional. Default: "title asc" string of field name and sort-order. The allowed fields are 'name', 'package_count' and 'title'
- **limit** (*int*) – the maximum number of organizations returned (optional) Default: `1000` when all_fields=false unless set in site's configuration `ckan.group_and_organization_list_max` Default: `25` when all_fields=true unless set in site's configuration `ckan.group_and_organization_list_all_fields_max`
- **offset** (*int*) – when `limit` is given, the offset to start returning organizations from
- **organizations** (*list of strings*) – a list of names of the groups to return, if given only groups whose names are in this list will be returned (optional)
- **all_fields** (*bool*) – return group dictionaries instead of just names. Only core fields are returned - get some more using the include_* options. Returning a list of packages is

too expensive, so the *packages* property for each group is deprecated, but there is a count of the packages in the *package_count* property. (optional, default: `False`)

- **include_dataset_count** (*bool*) – if all_fields, include the full package_count (optional, default: `True`)
- **include_extras** (*bool*) – if all_fields, include the organization extra fields (optional, default: `False`)
- **include_tags** (*bool*) – if all_fields, include the organization tags (optional, default: `False`)
- **include_groups** (*bool*) – if all_fields, include the organizations the organizations are in (optional, default: `False`)
- **include_users** (*bool*) – if all_fields, include the organization users (optional, default: `False`).

Return type list of strings

ckan.logic.action.get.**group_list_authz**(*context*, *data_dict*)

Return the list of groups that the user is authorized to edit.

Parameters

- **available_only** (*bool*) – remove the existing groups in the package (optional, default: `False`)
- **am_member** (*bool*) – if `True` return only the groups the logged-in user is a member of, otherwise return all groups that the user is authorized to edit (for example, sysadmin users are authorized to edit all groups) (optional, default: `False`)

Returns list of dictized groups that the user is authorized to edit

Return type list of dicts

ckan.logic.action.get.**organization_list_for_user**(*context*, *data_dict*)

Return the organizations that the user has a given permission for.

Specifically it returns the list of organizations that the currently authorized user has a given permission (for example: "manage_group") against.

By default this returns the list of organizations that the currently authorized user is member of, in any capacity.

When a user becomes a member of an organization in CKAN they're given a "capacity" (sometimes called a "role"), for example "member", "editor" or "admin".

Each of these roles has certain permissions associated with it. For example the admin role has the "admin" permission (which means they have permission to do anything). The editor role has permissions like "create_dataset", "update_dataset" and "delete_dataset". The member role has the "read" permission.

This function returns the list of organizations that the authorized user has a given permission for. For example the list of organizations that the user is an admin of, or the list of organizations that the user can create datasets in. This takes account of when permissions cascade down an organization hierarchy.

Parameters

- **id** (*string*) – the name or id of the user to get the organization list for (optional, defaults to the currently authorized user (logged in or via API key))
- **permission** (*string*) – the permission the user has against the returned organizations, for example `"read"` or `"create_dataset"` (optional, default: `"manage_group"`)
- **include_dataset_count** (*bool*) – include the package_count in each org (optional, default: `False`)

Returns list of organizations that the user has the given permission for

Return type list of dicts

ckan.logic.action.get.**license_list**(*context*, *data_dict*)
Return the list of licenses available for datasets on the site.

Return type list of dictionaries

ckan.logic.action.get.**tag_list**(*context*, *data_dict*)
Return a list of the site's tags.

By default only free tags (tags that don't belong to a vocabulary) are returned. If the `vocabulary_id` argument is given then only tags belonging to that vocabulary will be returned instead.

Parameters

- **query** (*string*) – a tag name query to search for, if given only tags whose names contain this string will be returned (optional)
- **vocabulary_id** (*string*) – the id or name of a vocabulary, if give only tags that belong to this vocabulary will be returned (optional)
- **all_fields** (*bool*) – return full tag dictionaries instead of just names (optional, default: `False`)

Return type list of dictionaries

ckan.logic.action.get.**user_list**(*context*, *data_dict*)
Return a list of the site's user accounts.

Parameters

- **q** (*string*) – filter the users returned to those whose names contain a string (optional)
- **email** (*string*) – filter the users returned to those whose email match a string (optional) (you must be a sysadmin to use this filter)
- **order_by** (*string*) – which field to sort the list by (optional, default: `'display_name'`). Users can be sorted by `'id'`, `'name'`, `'fullname'`, `'display_name'`, `'created'`, `'about'`, `'sysadmin'` or `'number_created_packages'`.
- **all_fields** (*bool*) – return full user dictionaries instead of just names. (optional, default: `True`)

Return type list of user dictionaries. User properties include: `number_created_packages` which excludes datasets which are private or draft state.

ckan.logic.action.get.**package_relationships_list**(*context*, *data_dict*)
Return a dataset (package)'s relationships.

Parameters

- **id** (*string*) – the id or name of the first package
- **id2** (*string*) – the id or name of the second package
- **rel** – relationship as string see *package_relationship_create()* for the relationship types (optional)

Return type list of dictionaries

ckan.logic.action.get.**package_show**(*context*, *data_dict*)
Return the metadata of a dataset (package) and its resources.

Parameters

- **id** (*string*) – the id or name of the dataset
- **use_default_schema** (*bool*) – use default package schema instead of a custom schema defined with an IDatasetForm plugin (default: `False`)
- **include_tracking** (*bool*) – add tracking information to dataset and resources (default: `False`)

Return type dictionary

ckan.logic.action.get.**resource_show**(*context*, *data_dict*)
Return the metadata of a resource.

Parameters

- **id** (*string*) – the id of the resource
- **include_tracking** (*bool*) – add tracking information to dataset and resources (default: `False`)

Return type dictionary

ckan.logic.action.get.**resource_view_show**(*context*, *data_dict*)
Return the metadata of a resource_view.

Parameters **id** (*string*) – the id of the resource_view

Return type dictionary

ckan.logic.action.get.**resource_view_list**(*context*, *data_dict*)
Return the list of resource views for a particular resource.

Parameters **id** (*string*) – the id of the resource

Return type list of dictionaries.

ckan.logic.action.get.**group_show**(*context*, *data_dict*)
Return the details of a group.

Parameters

- **id** (*string*) – the id or name of the group
- **include_datasets** (*bool*) – include a truncated list of the group's datasets (optional, default: `False`)
- **include_dataset_count** (*bool*) – include the full package_count (optional, default: `True`)
- **include_extras** (*bool*) – include the group's extra fields (optional, default: `True`)
- **include_users** (*bool*) – include the group's users (optional, default: `True` if `ckan.auth.public_user_details` is `True` otherwise `False`)
- **include_groups** (*bool*) – include the group's sub groups (optional, default: `True`)
- **include_tags** (*bool*) – include the group's tags (optional, default: `True`)
- **include_followers** (*bool*) – include the group's number of followers (optional, default: `True`)

Return type dictionary

Note: Only its first 1000 datasets are returned

ckan.logic.action.get.**organization_show**(*context*, *data_dict*)
Return the details of a organization.

Parameters

- **id** (*string*) – the id or name of the organization
- **include_datasets** (*bool*) – include a truncated list of the org's datasets (optional, default: `False`)
- **include_dataset_count** (*bool*) – include the full package_count (optional, default: `True`)
- **include_extras** (*bool*) – include the organization's extra fields (optional, default: `True`)
- **include_users** (*bool*) – include the organization's users (optional, default: `True` if `ckan.auth.public_user_details` is `True` otherwise `False`)
- **include_groups** (*bool*) – include the organization's sub groups (optional, default: `True`)
- **include_tags** (*bool*) – include the organization's tags (optional, default: `True`)
- **include_followers** (*bool*) – include the organization's number of followers (optional, default: `True`)

Return type dictionary

Note: Only its first 10 datasets are returned

ckan.logic.action.get.**group_package_show**(*context*, *data_dict*)
Return the datasets (packages) of a group.

Parameters

- **id** (*string*) – the id or name of the group
- **limit** (*int*) – the maximum number of datasets to return (optional)

Return type list of dictionaries

ckan.logic.action.get.**tag_show**(*context*, *data_dict*)
Return the details of a tag and all its datasets.

Parameters

- **id** (*string*) – the name or id of the tag
- **vocabulary_id** (*string*) – the id or name of the tag vocabulary that the tag is in - if it is not specified it will assume it is a free tag. (optional)
- **include_datasets** (*bool*) – include a list of the tag's datasets. (Up to a limit of 1000 - for more flexibility, use package_search - see *package_search()* for an example.) (optional, default: `False`)

Returns the details of the tag, including a list of all of the tag's datasets and their details

Return type dictionary

ckan.logic.action.get.**user_show**(*context*, *data_dict*)
Return a user account.

Either the `id` or the `user_obj` parameter must be given.

Parameters

- **id** (*string*) – the id or name of the user (optional)
- **user_obj** (*user dictionary*) – the user dictionary of the user (optional)
- **include_datasets** (*bool*) – Include a list of datasets the user has created. If it is the same user or a sysadmin requesting, it includes datasets that are draft or private. (optional, default:`False`, limit:50)
- **include_num_followers** (*bool*) – Include the number of followers the user has (optional, default:`False`)
- **include_password_hash** (*bool*) – Include the stored password hash (sysadmin only, optional, default:`False`)
- **include_plugin_extras** (*bool*) – Include the internal plugin extras object (sysadmin only, optional, default:`False`)

Returns the details of the user. Includes email_hash and number_created_packages (which excludes draft or private datasets unless it is the same user or sysadmin making the request). Excludes the password (hash) and reset_key. If it is the same user or a sysadmin requesting, the email and apikey are included.

Return type dictionary

ckan.logic.action.get.**package_autocomplete**(*context*, *data_dict*)
Return a list of datasets (packages) that match a string.

Datasets with names or titles that contain the query string will be returned.

Parameters

- **q** (*string*) – the string to search for
- **limit** (*int*) – the maximum number of resource formats to return (optional, default: `10`)

Return type list of dictionaries

ckan.logic.action.get.**format_autocomplete**(*context*, *data_dict*)
Return a list of resource formats whose names contain a string.

Parameters

- **q** (*string*) – the string to search for
- **limit** (*int*) – the maximum number of resource formats to return (optional, default: `5`)

Return type list of strings

ckan.logic.action.get.**user_autocomplete**(*context*, *data_dict*)
Return a list of user names that contain a string.

Parameters

- **q** (*string*) – the string to search for
- **limit** (*int*) – the maximum number of user names to return (optional, default: `20`)

Return type a list of user dictionaries each with keys `'name'`, `'fullname'`, and `'id'`

ckan.logic.action.get.**group_autocomplete**(*context*, *data_dict*)
Return a list of group names that contain a string.

Parameters

- **q** (*string*) – the string to search for
- **limit** (*int*) – the maximum number of groups to return (optional, default: 20)

Return type a list of group dictionaries each with keys `'name'`, `'title'`, and `'id'`

ckan.logic.action.get.**organization_autocomplete**(*context*, *data_dict*)
Return a list of organization names that contain a string.

Parameters

- **q** (*string*) – the string to search for
- **limit** (*int*) – the maximum number of organizations to return (optional, default: `20`)

Return type a list of organization dictionaries each with keys `'name'`, `'title'`, and `'id'`

ckan.logic.action.get.**package_search**(*context*, *data_dict*)
Searches for packages satisfying a given search criteria.

This action accepts solr search query parameters (details below), and returns a dictionary of results, including dictized datasets that match the search criteria, a search count and also facet information.

Solr Parameters:

For more in depth treatment of each paramter, please read the Solr Documentation.

This action accepts a *subset* of solr's search query parameters:

Parameters

- **q** (*string*) – the solr query. Optional. Default: `"*:*"`
- **fq** (*string*) – any filter queries to apply. Note: `+site_id:{ckan_site_id}` is added to this string prior to the query being executed.
- **fq_list** (*list of strings*) – additional filter queries to apply.
- **sort** (*string*) – sorting of the search results. Optional. Default: `'score desc, metadata_modified desc'`. As per the solr documentation, this is a comma-separated string of field names and sort-orderings.
- **rows** (*int*) – the maximum number of matching rows (datasets) to return. (optional, default: `10`, upper limit: `1000` unless set in site's configuration `ckan.search.rows_max`)
- **start** (*int*) – the offset in the complete result for where the set of returned datasets should begin.
- **facet** (*string*) – whether to enable faceted results. Default: `True`.
- **facet.mincount** (*int*) – the minimum counts for facet fields should be included in the results.
- **facet.limit** (*int*) – the maximum number of values the facet fields return. A negative value means unlimited. This can be set instance-wide with the *search.facets.limit* config option. Default is 50.
- **facet.field**(*list of strings*) – the fields to facet upon. Default empty. If empty, then the returned facet information is empty.

- **include_drafts** (*bool*) – if `True`, draft datasets will be included in the results. A user will only be returned their own draft datasets, and a sysadmin will be returned all draft datasets. Optional, the default is `False`.
- **include_private** (*bool*) – if `True`, private datasets will be included in the results. Only private datasets from the user's organizations will be returned and sysadmins will be returned all private datasets. Optional, the default is `False`.
- **use_default_schema** (*bool*) – use default package schema instead of a custom schema defined with an IDatasetForm plugin (default: `False`)

The following advanced Solr parameters are supported as well. Note that some of these are only available on particular Solr versions. See Solr's dismax and edismax documentation for further details on them:

`qf`, `wt`, `bf`, `boost`, `tie`, `defType`, `mm`

Examples:

`q=flood` datasets containing the word *flood*, *floods* or *flooding* `fq=tags:economy` datasets with the tag *economy* `facet.field=["tags"] facet.limit=10 rows=0` top 10 tags

Results:

The result of this action is a dict with the following keys:

Return type A dictionary with the following keys

Parameters

- **count** (*int*) – the number of results found. Note, this is the total number of results found, not the total number of results returned (which is affected by limit and row parameters used in the input).
- **results** (*list of dictized datasets.*) – ordered list of datasets matching the query, where the ordering defined by the sort parameter used in the query.
- **facets** (*DEPRECATED dict*) – DEPRECATED. Aggregated information about facet counts.
- **search_facets** (*nested dict of dicts.*) – aggregated information about facet counts. The outer dict is keyed by the facet field name (as used in the search query). Each entry of the outer dict is itself a dict, with a "title" key, and an "items" key. The "items" key's value is a list of dicts, each with "count", "display_name" and "name" entries. The display_name is a form of the name that can be used in titles.

An example result:

```
{'count': 2,
 'results': [ { <snip> }, { <snip> }],
 'search_facets': {u'tags': {'items': [{'count': 1,
                                        'display_name': u'tolstoy',
                                        'name': u'tolstoy'},
                                       {'count': 2,
                                        'display_name': u'russian',
                                        'name': u'russian'}
                                      ]
                            }
                   }
}
```

Limitations:

The full solr query language is not exposed, including.

fl The parameter that controls which fields are returned in the solr query. fl can be None or a list of result fields, such as ['id', 'extras_custom_field']. if fl = None, datasets are returned as a list of full dictionary.

ckan.logic.action.get.**resource_search**(*context*, *data_dict*)

Searches for resources satisfying a given search criteria.

It returns a dictionary with 2 fields: `count` and `results`. The `count` field contains the total number of Resources found without the limit or query parameters having an effect. The `results` field is a list of dictized Resource objects.

The 'query' parameter is a required field. It is a string of the form `{field}:{term}` or a list of strings, each of the same form. Within each string, `{field}` is a field or extra field on the Resource domain object.

If `{field}` is `"hash"`, then an attempt is made to match the *{term}* as a *prefix* of the `Resource.hash` field.

If `{field}` is an extra field, then an attempt is made to match against the extra fields stored against the Resource.

Note: The search is limited to search against extra fields declared in the config setting `ckan.extra_resource_fields`.

Note: Due to a Resource's extra fields being stored as a json blob, the match is made against the json string representation. As such, false positives may occur:

If the search criteria is:

```
query = "field1:term1"
```

Then a json blob with the string representation of:

```
{"field1": "foo", "field2": "term1"}
```

will match the search criteria! This is a known short-coming of this approach.

All matches are made ignoring case; and apart from the `"hash"` field, a term matches if it is a substring of the field's value.

Finally, when specifying more than one search criteria, the criteria are AND-ed together.

The `order` parameter is used to control the ordering of the results. Currently only ordering one field is available, and in ascending order only.

The `fields` parameter is deprecated as it is not compatible with calling this action with a GET request to the action API.

The context may contain a flag, *search_query*, which if True will make this action behave as if being used by the internal search api. ie - the results will not be dictized, and SearchErrors are thrown for bad search queries (rather than ValidationErrors).

Parameters

- **query** (string or list of strings of the form `{field}:{term1}`) – The search criteria. See above for description.
- **fields** (*dict of fields to search terms.*) – Deprecated
- **order_by** (*string*) – A field on the Resource model that orders the results.
- **offset** (*int*) – Apply an offset to the query.
- **limit** (*int*) – Apply a limit to the query.

Returns A dictionary with a `count` field, and a `results` field.

Return type dict

ckan.logic.action.get.**tag_search**(*context*, *data_dict*)
Return a list of tags whose names contain a given string.

By default only free tags (tags that don't belong to any vocabulary) are searched. If the `vocabulary_id` argument is given then only tags belonging to that vocabulary will be searched instead.

Parameters

- **query** (*string or list of strings*) – the string(s) to search for
- **vocabulary_id** (*string*) – the id or name of the tag vocabulary to search in (optional)
- **fields** (*dictionary*) – deprecated
- **limit** (*int*) – the maximum number of tags to return
- **offset** (*int*) – when `limit` is given, the offset to start returning tags from

Returns

A dictionary with the following keys:

'count' The number of tags in the result.

'results' The list of tags whose names contain the given string, a list of dictionaries.

Return type dictionary

ckan.logic.action.get.**tag_autocomplete**(*context*, *data_dict*)
Return a list of tag names that contain a given string.

By default only free tags (tags that don't belong to any vocabulary) are searched. If the `vocabulary_id` argument is given then only tags belonging to that vocabulary will be searched instead.

Parameters

- **query** (*string*) – the string to search for
- **vocabulary_id** (*string*) – the id or name of the tag vocabulary to search in (optional)
- **fields** (*dictionary*) – deprecated
- **limit** (*int*) – the maximum number of tags to return
- **offset** (*int*) – when `limit` is given, the offset to start returning tags from

Return type list of strings

ckan.logic.action.get.**task_status_show**(*context*, *data_dict*)
Return a task status.

Either the `id` parameter *or* the `entity_id`, `task_type` *and* `key` parameters must be given.

Parameters

- **id** (*string*) – the id of the task status (optional)
- **entity_id** (*string*) – the entity_id of the task status (optional)
- **task_type** (*string*) – the task_type of the task status (optional)
- **key** (*string*) – the key of the task status (optional)

Return type dictionary

ckan.logic.action.get.**term_translation_show**(*context*, *data_dict*)
Return the translations for the given term(s) and language(s).

Parameters

- **terms** (*list of strings*) – the terms to search for translations of, e.g. `'Russian'`, `'romantic novel'`
- **lang_codes** (*list of language code strings*) – the language codes of the languages to search for translations into, e.g. `'en'`, `'de'` (optional, default is to search for translations into any language)

Return type a list of term translation dictionaries each with keys `'term'` (the term searched for, in the source language), `'term_translation'` (the translation of the term into the target language) and `'lang_code'` (the language code of the target language)

ckan.logic.action.get.**get_site_user**(*context*, *data_dict*)
Return the ckan site user

Parameters **defer_commit** (*bool*) – by default (or if set to false) get_site_user will commit and clean up the current transaction. If set to true, caller is responsible for commiting transaction after get_site_user is called. Leaving open connections can cause cli commands to hang! (optional, default: `False`)

ckan.logic.action.get.**status_show**(*context*, *data_dict*)
Return a dictionary with information about the site's configuration.

Return type dictionary

ckan.logic.action.get.**vocabulary_list**(*context*, *data_dict*)
Return a list of all the site's tag vocabularies.

Return type list of dictionaries

ckan.logic.action.get.**vocabulary_show**(*context*, *data_dict*)
Return a single tag vocabulary.

Parameters **id** (*string*) – the id or name of the vocabulary

Returns the vocabulary.

Return type dictionary

ckan.logic.action.get.**user_activity_list**(*context*, *data_dict*)
Return a user's public activity stream.

You must be authorized to view the user's profile.

Parameters

- **id** (*string*) – the id or name of the user
- **offset** (*int*) – where to start getting activity items from (optional, default: `0`)
- **limit** (*int*) – the maximum number of activities to return (optional, default: `31` unless set in site's configuration `ckan.activity_list_limit`, upper limit: `100` unless set in site's configuration `ckan.activity_list_limit_max`)

Return type list of dictionaries

ckan.logic.action.get.**package_activity_list**(*context*, *data_dict*)
Return a package's activity stream (not including detail)

You must be authorized to view the package.

Parameters

- **id** (*string*) – the id or name of the package
- **offset** (*int*) – where to start getting activity items from (optional, default: `0`)

- **limit** (*int*) – the maximum number of activities to return (optional, default: 31 unless set in site's configuration ckan.activity_list_limit, upper limit: 100 unless set in site's configuration ckan.activity_list_limit_max)
- **include_hidden_activity** (*bool*) – whether to include 'hidden' activity, which is not shown in the Activity Stream page. Hidden activity includes activity done by the site_user, such as harvests, which are not shown in the activity stream because they can be too numerous, or activity by other users specified in config option *ckan.hide_activity_from_users*. NB Only sysadmins may set include_hidden_activity to true. (default: false)

Return type list of dictionaries

ckan.logic.action.get.**group_activity_list**(*context*, *data_dict*)

Return a group's activity stream.

You must be authorized to view the group.

Parameters

- **id** (*string*) – the id or name of the group
- **offset** (*int*) – where to start getting activity items from (optional, default: 0)
- **limit** (*int*) – the maximum number of activities to return (optional, default: 31 unless set in site's configuration ckan.activity_list_limit, upper limit: 100 unless set in site's configuration ckan.activity_list_limit_max)
- **include_hidden_activity** (*bool*) – whether to include 'hidden' activity, which is not shown in the Activity Stream page. Hidden activity includes activity done by the site_user, such as harvests, which are not shown in the activity stream because they can be too numerous, or activity by other users specified in config option *ckan.hide_activity_from_users*. NB Only sysadmins may set include_hidden_activity to true. (default: false)

Return type list of dictionaries

ckan.logic.action.get.**organization_activity_list**(*context*, *data_dict*)

Return a organization's activity stream.

Parameters

- **id** (*string*) – the id or name of the organization
- **offset** (*int*) – where to start getting activity items from (optional, default: 0)
- **limit** (*int*) – the maximum number of activities to return (optional, default: 31 unless set in site's configuration ckan.activity_list_limit, upper limit: 100 unless set in site's configuration ckan.activity_list_limit_max)
- **include_hidden_activity** (*bool*) – whether to include 'hidden' activity, which is not shown in the Activity Stream page. Hidden activity includes activity done by the site_user, such as harvests, which are not shown in the activity stream because they can be too numerous, or activity by other users specified in config option *ckan.hide_activity_from_users*. NB Only sysadmins may set include_hidden_activity to true. (default: false)

Return type list of dictionaries

ckan.logic.action.get.**recently_changed_packages_activity_list**(*context*, *data_dict*)

Return the activity stream of all recently added or changed packages.

Parameters

- **offset** (*int*) – where to start getting activity items from (optional, default: 0)
- **limit** (*int*) – the maximum number of activities to return (optional, default: 31 unless set in site's configuration ckan.activity_list_limit, upper limit: 100 unless set in site's configuration ckan.activity_list_limit_max)

Return type list of dictionaries

ckan.logic.action.get.**user_follower_count**(*context*, *data_dict*)
Return the number of followers of a user.

Parameters **id** (*string*) – the id or name of the user

Return type int

ckan.logic.action.get.**dataset_follower_count**(*context*, *data_dict*)
Return the number of followers of a dataset.

Parameters **id** (*string*) – the id or name of the dataset

Return type int

ckan.logic.action.get.**group_follower_count**(*context*, *data_dict*)
Return the number of followers of a group.

Parameters **id** (*string*) – the id or name of the group

Return type int

ckan.logic.action.get.**organization_follower_count**(*context*, *data_dict*)
Return the number of followers of an organization.

Parameters **id** (*string*) – the id or name of the organization

Return type int

ckan.logic.action.get.**user_follower_list**(*context*, *data_dict*)
Return the list of users that are following the given user.

Parameters **id** (*string*) – the id or name of the user

Return type list of dictionaries

ckan.logic.action.get.**dataset_follower_list**(*context*, *data_dict*)
Return the list of users that are following the given dataset.

Parameters **id** (*string*) – the id or name of the dataset

Return type list of dictionaries

ckan.logic.action.get.**group_follower_list**(*context*, *data_dict*)
Return the list of users that are following the given group.

Parameters **id** (*string*) – the id or name of the group

Return type list of dictionaries

ckan.logic.action.get.**organization_follower_list**(*context*, *data_dict*)
Return the list of users that are following the given organization.

Parameters **id** (*string*) – the id or name of the organization

Return type list of dictionaries

ckan.logic.action.get.**am_following_user**(*context*, *data_dict*)
Return True if you're following the given user, False if not.

Parameters **id** (*string*) – the id or name of the user

Return type bool

ckan.logic.action.get.**am_following_dataset**(*context*, *data_dict*)
Return `True` if you're following the given dataset, `False` if not.

Parameters **id** (*string*) – the id or name of the dataset

Return type bool

ckan.logic.action.get.**am_following_group**(*context*, *data_dict*)
Return `True` if you're following the given group, `False` if not.

Parameters **id** (*string*) – the id or name of the group

Return type bool

ckan.logic.action.get.**followee_count**(*context*, *data_dict*)
Return the number of objects that are followed by the given user.

Counts all objects, of any type, that the given user is following (e.g. followed users, followed datasets, followed groups).

Parameters **id** (*string*) – the id of the user

Return type int

ckan.logic.action.get.**user_followee_count**(*context*, *data_dict*)
Return the number of users that are followed by the given user.

Parameters **id** (*string*) – the id of the user

Return type int

ckan.logic.action.get.**dataset_followee_count**(*context*, *data_dict*)
Return the number of datasets that are followed by the given user.

Parameters **id** (*string*) – the id of the user

Return type int

ckan.logic.action.get.**group_followee_count**(*context*, *data_dict*)
Return the number of groups that are followed by the given user.

Parameters **id** (*string*) – the id of the user

Return type int

ckan.logic.action.get.**followee_list**(*context*, *data_dict*)
Return the list of objects that are followed by the given user.

Returns all objects, of any type, that the given user is following (e.g. followed users, followed datasets, followed groups..).

Parameters

- **id** (*string*) – the id of the user
- **q** (*string*) – a query string to limit results by, only objects whose display name begins with the given string (case-insensitive) wil be returned (optional)

Return type list of dictionaries, each with keys `'type'` (e.g. `'user'`, `'dataset'` or `'group'`), `'display_name'` (e.g. a user's display name, or a package's title) and `'dict'` (e.g. a dict representing the followed user, package or group, the same as the dict that would be returned by *user_show()*, *package_show()* or *group_show()*)

ckan.logic.action.get.**user_followee_list**(*context*, *data_dict*)
Return the list of users that are followed by the given user.

Parameters **id** (*string*) – the id of the user

Return type list of dictionaries

ckan.logic.action.get.**dataset_followee_list**(*context*, *data_dict*)
Return the list of datasets that are followed by the given user.

Parameters **id** (*string*) – the id or name of the user

Return type list of dictionaries

ckan.logic.action.get.**group_followee_list**(*context*, *data_dict*)
Return the list of groups that are followed by the given user.

Parameters **id** (*string*) – the id or name of the user

Return type list of dictionaries

ckan.logic.action.get.**organization_followee_list**(*context*, *data_dict*)
Return the list of organizations that are followed by the given user.

Parameters **id** (*string*) – the id or name of the user

Return type list of dictionaries

ckan.logic.action.get.**dashboard_activity_list**(*context*, *data_dict*)

Return the authorized (via login or API key) user's dashboard activity stream.

Unlike the activity dictionaries returned by other `*_activity_list` actions, these activity dictionaries have an extra boolean value with key `is_new` that tells you whether the activity happened since the user last viewed her dashboard (`'is_new': True`) or not (`'is_new': False`).

The user's own activities are always marked `'is_new': False`.

Parameters

- **offset** (*int*) – where to start getting activity items from (optional, default: `0`)
- **limit** (*int*) – the maximum number of activities to return (optional, default: `31` unless set in site's configuration `ckan.activity_list_limit`, upper limit: `100` unless set in site's configuration `ckan.activity_list_limit_max`)

Return type list of activity dictionaries

ckan.logic.action.get.**dashboard_new_activities_count**(*context*, *data_dict*)
Return the number of new activities in the user's dashboard.

Return the number of new activities in the authorized user's dashboard activity stream.

Activities from the user herself are not counted by this function even though they appear in the dashboard (users don't want to be notified about things they did themselves).

Return type int

ckan.logic.action.get.**activity_show**(*context*, *data_dict*)
Show details of an item of 'activity' (part of the activity stream).

Parameters

- **id** (*string*) – the id of the activity
- **include_data** (*boolean*) – include the data field, containing a full object dict (otherwise the data field is only returned with the object's title)

Return type dictionary

ckan.logic.action.get.**activity_data_show**(*context*, *data_dict*)
Show the data from an item of 'activity' (part of the activity stream).

For example for a package update this returns just the dataset dict but none of the activity stream info of who and when the version was created.

Parameters

- **id** (*string*) – the id of the activity
- **object_type** (*string*) – 'package', 'user', 'group' or 'organization'

Return type dictionary

ckan.logic.action.get.**activity_diff**(*context*, *data_dict*)
Returns a diff of the activity, compared to the previous version of the object

Parameters

- **id** (*string*) – the id of the activity
- **object_type** (*string*) – 'package', 'user', 'group' or 'organization'
- **diff_type** (*string*) – 'unified', 'context', 'html'

ckan.logic.action.get.**member_roles_list**(*context*, *data_dict*)
Return the possible roles for members of groups and organizations.

Parameters **group_type** (*string*) – the group type, either `"group"` or `"organization"` (optional, default `"organization"`)

Returns a list of dictionaries each with two keys: `"text"` (the display name of the role, e.g. `"Admin"`) and `"value"` (the internal name of the role, e.g. `"admin"`)

Return type list of dictionaries

ckan.logic.action.get.**help_show**(*context*, *data_dict*)
Return the help string for a particular API action.

Parameters **name** (*string*) – Action function name (eg *user_create*, *package_search*)

Returns The help string for the action function, or None if the function does not have a docstring.

Return type string

Raises `ckan.logic.NotFound`: if the action function doesn't exist

ckan.logic.action.get.**config_option_show**(*context*, *data_dict*)
Show the current value of a particular configuration option.

Only returns runtime-editable config options (the ones returned by *config_option_list()*), which can be updated with the *config_option_update()* action.

Parameters **key** (*string*) – The configuration option key

Returns The value of the config option from either the system_info table or ini file.

Return type string

Raises `ckan.logic.ValidationError`: if config option is not in the schema (whitelisted as editable).

ckan.logic.action.get.**config_option_list**(*context*, *data_dict*)

Return a list of runtime-editable config options keys that can be updated with *config_option_update()*.

Returns A list of config option keys.

Return type list

ckan.logic.action.get.**job_list**(*context*, *data_dict*)
List enqueued background jobs.

Parameters **queues** (*list*) – Queues to list jobs from. If not given then the jobs from all queues are listed.

Returns The currently enqueued background jobs.

Return type list

New in version 2.7.

ckan.logic.action.get.**job_show**(*context*, *data_dict*)
Show details for a background job.

Parameters **id** (*string*) – The ID of the background job.

Returns Details about the background job.

Return type dict

New in version 2.7.

ckan.logic.action.get.**api_token_list**(*context*, *data_dict*)
Return list of all available API Tokens for current user.

Returns collection of all API Tokens

Return type list

New in version 2.9.

4.9.2 ckan.logic.action.create

API functions for adding data to CKAN.

ckan.logic.action.create.**package_create**(*context*, *data_dict*)
Create a new dataset (package).

You must be authorized to create new datasets. If you specify any groups for the new dataset, you must also be authorized to edit these groups.

Plugins may change the parameters of this function depending on the value of the `type` parameter, see the *IDatasetForm* plugin interface.

Parameters

- **name** (*string*) – the name of the new dataset, must be between 2 and 100 characters long and contain only lowercase alphanumeric characters, - and _, e.g. `'warandpeace'`
- **title** (*string*) – the title of the dataset (optional, default: same as `name`)
- **private** (*bool*) – If `True` creates a private dataset
- **author** (*string*) – the name of the dataset's author (optional)
- **author_email** (*string*) – the email address of the dataset's author (optional)
- **maintainer** (*string*) – the name of the dataset's maintainer (optional)
- **maintainer_email** (*string*) – the email address of the dataset's maintainer (optional)

- **license_id** (*license id string*) – the id of the dataset's license, see *license_list()* for available values (optional)
- **notes** (*string*) – a description of the dataset (optional)
- **url** (*string*) – a URL for the dataset's source (optional)
- **version** (*string, no longer than 100 characters*) – (optional)
- **state** (*string*) – the current state of the dataset, e.g. `'active'` or `'deleted'`, only active datasets show up in search results and other lists of datasets, this parameter will be ignored if you are not authorized to change the state of the dataset (optional, default: `'active'`)
- **type** (*string*) – the type of the dataset (optional), *IDatasetForm* plugins associate themselves with different dataset types and provide custom dataset handling behaviour for these types
- **resources** (*list of resource dictionaries*) – the dataset's resources, see *resource_create()* for the format of resource dictionaries (optional)
- **tags** (*list of tag dictionaries*) – the dataset's tags, see *tag_create()* for the format of tag dictionaries (optional)
- **extras** (*list of dataset extra dictionaries*) – the dataset's extras (optional), extras are arbitrary (key: value) metadata items that can be added to datasets, each extra dictionary should have keys `'key'` (a string), `'value'` (a string)
- **relationships_as_object** (*list of relationship dictionaries*) – see *package_relationship_create()* for the format of relationship dictionaries (optional)
- **relationships_as_subject** (*list of relationship dictionaries*) – see *package_relationship_create()* for the format of relationship dictionaries (optional)
- **groups** (*list of dictionaries*) – the groups to which the dataset belongs (optional), each group dictionary should have one or more of the following keys which identify an existing group: `'id'` (the id of the group, string), or `'name'` (the name of the group, string), to see which groups exist call *group_list()*
- **owner_org** (*string*) – the id of the dataset's owning organization, see *organization_list()* or *organization_list_for_user()* for available values. This parameter can be made optional if the config option *ckan.auth.create_unowned_dataset* is set to `True`.

Returns the newly created dataset (unless 'return_id_only' is set to True in the context, in which case just the dataset id will be returned)

Return type dictionary

ckan.logic.action.create.**resource_create**(*context*, *data_dict*)

Appends a new resource to a datasets list of resources.

Parameters

- **package_id** (*string*) – id of package that the resource should be added to.
- **url** (*string*) – url of resource
- **description** (*string*) – (optional)
- **format** (*string*) – (optional)

- **hash** (*string*) – (optional)
- **name** (*string*) – (optional)
- **resource_type** (*string*) – (optional)
- **mimetype** (*string*) – (optional)
- **mimetype_inner** (*string*) – (optional)
- **cache_url** (*string*) – (optional)
- **size** (*int*) – (optional)
- **created** (*iso date string*) – (optional)
- **last_modified** (*iso date string*) – (optional)
- **cache_last_updated** (*iso date string*) – (optional)
- **upload** (*FieldStorage (optional) needs multipart/form-data*) – (optional)

Returns the newly created resource

Return type dictionary

ckan.logic.action.create.**resource_view_create**(*context*, *data_dict*)

Creates a new resource view.

Parameters

- **resource_id** (*string*) – id of the resource
- **title** (*string*) – the title of the view
- **description** (*string*) – a description of the view (optional)
- **view_type** (*string*) – type of view
- **config** (*JSON string*) – options necessary to recreate a view state (optional)

Returns the newly created resource view

Return type dictionary

ckan.logic.action.create.**resource_create_default_resource_views**(*context*, *data_dict*)

Creates the default views (if necessary) on the provided resource

The function will get the plugins for the default views defined in the configuration, and if some were found the *can_view* method of each one of them will be called to determine if a resource view should be created. Resource views extensions get the resource dict and the parent dataset dict.

If the latter is not provided, *package_show* is called to get it.

By default only view plugins that don't require the resource data to be in the DataStore are called. See ckan.logic.action.create.package_create_default_resource_views.`()` for details on the create_datastore_views parameter.

Parameters

- **resource** (*dict*) – full resource dict
- **package** (*dict*) – full dataset dict (optional, if not provided *package_show()* will be called).
- **create_datastore_views** (*bool*) – whether to create views that rely on data being on the DataStore (optional, defaults to False)

Returns a list of resource views created (empty if none were created)

Return type list of dictionaries

ckan.logic.action.create.**package_create_default_resource_views**(*context*, *data_dict*)

Creates the default views on all resources of the provided dataset

By default only view plugins that don't require the resource data to be in the DataStore are called. Passing *create_datastore_views* as True will only create views that require data to be in the DataStore. The first case happens when the function is called from *package_create* or *package_update*, the second when it's called from the DataPusher when data was uploaded to the DataStore.

Parameters

- **package** (*dict*) – full dataset dict (ie the one obtained calling *package_show()*).
- **create_datastore_views** (*bool*) – whether to create views that rely on data being on the DataStore (optional, defaults to False)

Returns a list of resource views created (empty if none were created)

Return type list of dictionaries

ckan.logic.action.create.**package_relationship_create**(*context*, *data_dict*)

Create a relationship between two datasets (packages).

You must be authorized to edit both the subject and the object datasets.

Parameters

- **subject** (*string*) – the id or name of the dataset that is the subject of the relationship
- **object** – the id or name of the dataset that is the object of the relationship
- **type** (*string*) – the type of the relationship, one of `'depends_on'`, `'dependency_of'`, `'derives_from'`, `'has_derivation'`, `'links_to'`, `'linked_from'`, `'child_of'` or `'parent_of'`
- **comment** (*string*) – a comment about the relationship (optional)

Returns the newly created package relationship

Return type dictionary

ckan.logic.action.create.**member_create**(*context*, *data_dict=None*)

Make an object (e.g. a user, dataset or group) a member of a group.

If the object is already a member of the group then the capacity of the membership will be updated.

You must be authorized to edit the group.

Parameters

- **id** (*string*) – the id or name of the group to add the object to
- **object** (*string*) – the id or name of the object to add
- **object_type** (*string*) – the type of the object being added, e.g. `'package'` or `'user'`
- **capacity** (*string*) – the capacity of the membership

Returns the newly created (or updated) membership

Return type dictionary

ckan.logic.action.create.**package_collaborator_create**(*context*, *data_dict*)
Make a user a collaborator in a dataset.

If the user is already a collaborator in the dataset then their capacity will be updated.

Currently you must be an Admin on the dataset owner organization to manage collaborators.

Note: This action requires the collaborators feature to be enabled with the *ckan.auth.allow_dataset_collaborators* configuration option.

Parameters

- **id** (*string*) – the id or name of the dataset
- **user_id** (*string*) – the id or name of the user to add or edit
- **capacity** (*string*) – the capacity or role of the membership. Must be one of "editor" or "member". Additionally if *ckan.auth.allow_admin_collaborators* is set to True, "admin" is also allowed.

Returns the newly created (or updated) collaborator

Return type dictionary

ckan.logic.action.create.**group_create**(*context*, *data_dict*)
Create a new group.

You must be authorized to create groups.

Plugins may change the parameters of this function depending on the value of the `type` parameter, see the *IGroupForm* plugin interface.

Parameters

- **name** (*string*) – the name of the group, a string between 2 and 100 characters long, containing only lowercase alphanumeric characters, - and _
- **id** (*string*) – the id of the group (optional)
- **title** (*string*) – the title of the group (optional)
- **description** (*string*) – the description of the group (optional)
- **image_url** (*string*) – the URL to an image to be displayed on the group's page (optional)
- **type** (*string*) – the type of the group (optional, default: `'group'`), *IGroupForm* plugins associate themselves with different group types and provide custom group handling behaviour for these types Cannot be 'organization'
- **state** (*string*) – the current state of the group, e.g. `'active'` or `'deleted'`, only active groups show up in search results and other lists of groups, this parameter will be ignored if you are not authorized to change the state of the group (optional, default: `'active'`)
- **approval_status** (*string*) – (optional)
- **extras** (*list of dataset extra dictionaries*) – the group's extras (optional), extras are arbitrary (key: value) metadata items that can be added to groups, each extra dictionary should have keys `'key'` (a string), `'value'` (a string), and optionally `'deleted'`
- **packages** (*list of dictionaries*) – the datasets (packages) that belong to the group, a list of dictionaries each with keys `'name'` (string, the id or name of the dataset) and optionally `'title'` (string, the title of the dataset)

- **groups** (*list of dictionaries*) – the groups that belong to the group, a list of dictionaries each with key `'name'` (string, the id or name of the group) and optionally `'capacity'` (string, the capacity in which the group is a member of the group)
- **users** (*list of dictionaries*) – the users that belong to the group, a list of dictionaries each with key `'name'` (string, the id or name of the user) and optionally `'capacity'` (string, the capacity in which the user is a member of the group)

Returns the newly created group (unless 'return_id_only' is set to True in the context, in which case just the group id will be returned)

Return type dictionary

ckan.logic.action.create.**organization_create**(*context*, *data_dict*)

Create a new organization.

You must be authorized to create organizations.

Plugins may change the parameters of this function depending on the value of the `type` parameter, see the *IGroupForm* plugin interface.

Parameters

- **name** (*string*) – the name of the organization, a string between 2 and 100 characters long, containing only lowercase alphanumeric characters, - and _
- **id** (*string*) – the id of the organization (optional)
- **title** (*string*) – the title of the organization (optional)
- **description** (*string*) – the description of the organization (optional)
- **image_url** (*string*) – the URL to an image to be displayed on the organization's page (optional)
- **state** (*string*) – the current state of the organization, e.g. `'active'` or `'deleted'`, only active organizations show up in search results and other lists of organizations, this parameter will be ignored if you are not authorized to change the state of the organization (optional, default: `'active'`)
- **approval_status** (*string*) – (optional)
- **extras** (*list of dataset extra dictionaries*) – the organization's extras (optional), extras are arbitrary (key: value) metadata items that can be added to organizations, each extra dictionary should have keys `'key'` (a string), `'value'` (a string), and optionally `'deleted'`
- **packages** (*list of dictionaries*) – the datasets (packages) that belong to the organization, a list of dictionaries each with keys `'name'` (string, the id or name of the dataset) and optionally `'title'` (string, the title of the dataset)
- **users** (*list of dictionaries*) – the users that belong to the organization, a list of dictionaries each with key `'name'` (string, the id or name of the user) and optionally `'capacity'` (string, the capacity in which the user is a member of the organization)

Returns the newly created organization (unless 'return_id_only' is set to True in the context, in which case just the organization id will be returned)

Return type dictionary

ckan.logic.action.create.**rating_create**(*context*, *data_dict*)

Rate a dataset (package).

You must provide your API key in the Authorization header.

Parameters

- **package** (*string*) – the name or id of the dataset to rate
- **rating** (*int*) – the rating to give to the dataset, an integer between 1 and 5

Returns a dictionary with two keys: `'rating average'` (the average rating of the dataset you rated) and `'rating count'` (the number of times the dataset has been rated)

Return type dictionary

ckan.logic.action.create.**user_create**(*context*, *data_dict*)

Create a new user.

You must be authorized to create users.

Parameters

- **name** (*string*) – the name of the new user, a string between 2 and 100 characters in length, containing only lowercase alphanumeric characters, - and _
- **email** (*string*) – the email address for the new user
- **password** (*string*) – the password of the new user, a string of at least 4 characters
- **id** (*string*) – the id of the new user (optional)
- **fullname** (*string*) – the full name of the new user (optional)
- **about** (*string*) – a description of the new user (optional)
- **image_url** (*string*) – the URL to an image to be displayed on the group's page (optional)
- **plugin_extras** (*dict*) – private extra user data belonging to plugins. Only sysadmin users may set this value. It should be a dict that can be dumped into JSON, and plugins should namespace their extras with the plugin name to avoid collisions with other plugins, eg:

```
{
    "name": "test_user",
    "email": "test@example.com",
    "plugin_extras": {
        "my_plugin": {
            "private_extra": 1
        },
        "another_plugin": {
            "another_extra": True
        }
    }
}
```

Returns the newly created user

Return type dictionary

ckan.logic.action.create.**user_invite**(*context*, *data_dict*)

Invite a new user.

You must be authorized to create group members.

Parameters

- **email** (*string*) – the email of the user to be invited to the group
- **group_id** (*string*) – the id or name of the group

- **role** (*string*) – role of the user in the group. One of `member`, `editor`, or `admin`

Returns the newly created user

Return type dictionary

`ckan.logic.action.create.`**vocabulary_create**(*context*, *data_dict*)
Create a new tag vocabulary.

You must be a sysadmin to create vocabularies.

Parameters

- **name** (*string*) – the name of the new vocabulary, e.g. `'Genre'`
- **tags** (*list of tag dictionaries*) – the new tags to add to the new vocabulary, for the format of tag dictionaries see *tag_create()*

Returns the newly-created vocabulary

Return type dictionary

`ckan.logic.action.create.`**activity_create**(*context*, *activity_dict*, ***kw*)
Create a new activity stream activity.

You must be a sysadmin to create new activities.

Parameters

- **user_id** (*string*) – the name or id of the user who carried out the activity, e.g. `'seanh'`
- **object_id** – the name or id of the object of the activity, e.g. `'my_dataset'`
- **activity_type** (*string*) – the type of the activity, this must be an activity type that CKAN knows how to render, e.g. `'new package'`, `'changed user'`, `'deleted group'` etc.
- **data** (*dictionary*) – any additional data about the activity

Returns the newly created activity

Return type dictionary

`ckan.logic.action.create.`**tag_create**(*context*, *data_dict*)
Create a new vocabulary tag.

You must be a sysadmin to create vocabulary tags.

You can only use this function to create tags that belong to a vocabulary, not to create free tags. (To create a new free tag simply add the tag to a package, e.g. using the *package_update()* function.)

Parameters

- **name** (*string*) – the name for the new tag, a string between 2 and 100 characters long containing only alphanumeric characters and -, _ and `.`, e.g. `'Jazz'`
- **vocabulary_id** (*string*) – the id of the vocabulary that the new tag should be added to, e.g. the id of vocabulary `'Genre'`

Returns the newly-created tag

Return type dictionary

`ckan.logic.action.create.`**follow_user**(*context*, *data_dict*)
Start following another user.

You must provide your API key in the Authorization header.

Parameters **id** (*string*) – the id or name of the user to follow, e.g. `'joeuser'`

Returns a representation of the 'follower' relationship between yourself and the other user

Return type dictionary

ckan.logic.action.create.**follow_dataset**(*context*, *data_dict*)
Start following a dataset.

You must provide your API key in the Authorization header.

Parameters **id** (*string*) – the id or name of the dataset to follow, e.g. `'warandpeace'`

Returns a representation of the 'follower' relationship between yourself and the dataset

Return type dictionary

ckan.logic.action.create.**group_member_create**(*context*, *data_dict*)
Make a user a member of a group.

You must be authorized to edit the group.

Parameters

- **id** (*string*) – the id or name of the group
- **username** (*string*) – name or id of the user to be made member of the group
- **role** (*string*) – role of the user in the group. One of `member`, `editor`, or `admin`

Returns the newly created (or updated) membership

Return type dictionary

ckan.logic.action.create.**organization_member_create**(*context*, *data_dict*)
Make a user a member of an organization.

You must be authorized to edit the organization.

Parameters

- **id** (*string*) – the id or name of the organization
- **username** (*string*) – name or id of the user to be made member of the organization
- **role** (*string*) – role of the user in the organization. One of `member`, `editor`, or `admin`

Returns the newly created (or updated) membership

Return type dictionary

ckan.logic.action.create.**follow_group**(*context*, *data_dict*)
Start following a group.

You must provide your API key in the Authorization header.

Parameters **id** (*string*) – the id or name of the group to follow, e.g. `'roger'`

Returns a representation of the 'follower' relationship between yourself and the group

Return type dictionary

ckan.logic.action.create.**api_token_create**(*context*, *data_dict*)
Create new API Token for current user.

Apart from the *user* and *name* field that are required by default implementation, there may be additional fields registered by extensions.

Parameters

- **user** (*string*) – name or id of the user who owns new API Token
- **name** (*string*) – distinctive name for API Token

Returns Returns a dict with the key "token" containing the encoded token value. Extensions can privide additional fields via *add_extra* method of *IApiToken*

Return type dictionary

4.9.3 ckan.logic.action.update

API functions for updating existing data in CKAN.

ckan.logic.action.update.**resource_update**(*context*, *data_dict*)
Update a resource.

To update a resource you must be authorized to update the dataset that the resource belongs to.

Note: Update methods may delete parameters not explicitly provided in the data_dict. If you want to edit only a specific attribute use *resource_patch* instead.

For further parameters see *resource_create()*.

Parameters **id** (*string*) – the id of the resource to update

Returns the updated resource

Return type string

ckan.logic.action.update.**resource_view_update**(*context*, *data_dict*)
Update a resource view.

To update a resource_view you must be authorized to update the resource that the resource_view belongs to.

For further parameters see `resource_view_create()`.

Parameters **id** (*string*) – the id of the resource_view to update

Returns the updated resource_view

Return type string

ckan.logic.action.update.**resource_view_reorder**(*context*, *data_dict*)
Reorder resource views.

Parameters

- **id** (*string*) – the id of the resource
- **order** (*list of strings*) – the list of id of the resource to update the order of the views

Returns the updated order of the view

Return type dictionary

ckan.logic.action.update.**package_update**(*context*, *data_dict*)
Update a dataset (package).

You must be authorized to edit the dataset and the groups that it belongs to.

Note: Update methods may delete parameters not explicitly provided in the data_dict. If you want to edit only a specific attribute use *package_patch* instead.

It is recommended to call *ckan.logic.action.get.package_show()*, make the desired changes to the result, and then call `package_update()` with it.

Plugins may change the parameters of this function depending on the value of the dataset's `type` attribute, see the *IDatasetForm* plugin interface.

For further parameters see *package_create()*.

Parameters **id** (*string*) – the name or id of the dataset to update

Returns the updated dataset (if `'return_package_dict'` is `True` in the context, which is the default. Otherwise returns just the dataset id)

Return type dictionary

`ckan.logic.action.update.`**package_revise**(*context*, *data_dict*)

Revise a dataset (package) selectively with match, filter and update parameters.

You must be authorized to edit the dataset and the groups that it belongs to.

Parameters

- **match** (*dict*) – a dict containing "id" or "name" values of the dataset to update, all values provided must match the current dataset values or a ValidationError will be raised. e.g. `{"name": "my-data", "resources": {["name": "big.csv"]}}` would abort if the my-data dataset's first resource name is not "big.csv".
- **filter** (*comma-separated string patterns or list of string patterns*) – a list of string patterns of fields to remove from the current dataset. e.g. `"-resources__1"` would remove the second resource, `"+title, +resources, -*"` would remove all fields at the dataset level except title and all resources (default: `[]`)
- **update** (*dict*) – a dict with values to update/create after filtering e.g. `{"resources": [{"description": "file here"}]}` would update the description for the first resource
- **include** (*comma-separated-string patterns or list of string patterns*) – a list of string pattern of fields to include in response e.g. `"-*"` to return nothing (default: `[]` all fields returned)

`match` and `update` parameters may also be passed as "flattened keys", using either the item numeric index or its unique id (with a minimum of 5 characters), e.g. `update__resource__1f9ab__description="guidebook"` would set the description of the resource with id starting with "1f9ab" to "guidebook", and `update__resource__-1__description="guidebook"` would do the same on the last resource in the dataset.

The `extend` suffix can also be used on the update parameter to add a new item to a list, e.g. `update__resources__extend=[{"name": "new resource", "url": "https://example.com"}]` will add a new resource to the dataset with the provided `name` and `url`.

Usage examples:

- Change description in dataset, checking for old description:

```
match={"notes": "old notes", "name": "xyz"}
update={"notes": "new notes"}
```

- Identical to above, but using flattened keys:

```
match__name="xyz"
match__notes="old notes"
update__notes="new notes"
```

- Replace all fields at dataset level only, keep resources (note: dataset id and type fields can't be deleted)

```
match={"id": "1234abc-1420-cbad-1922"}
filter=["+resources", "-*"]
update={"name": "fresh-start", "title": "Fresh Start"}
```

- Add a new resource (__extend on flattened key):

```
match={"id": "abc0123-1420-cbad-1922"}
update__resources__extend=[{"name": "new resource", "url": "http://example.com
↪"}]
```

- Update a resource by its index:

```
match={"name": "my-data"}
update__resources__0={"name": "new name, first resource"}
```

- Update a resource by its id (provide at least 5 characters):

```
match={"name": "their-data"}
update__resources__19cfad={"description": "right one for sure"}
```

- Replace all fields of a resource:

```
match={"id": "34a12bc-1420-cbad-1922"}
filter=["+resources__1492a__id", "-resources__1492a__*"]
update__resources__1492a={"name": "edits here", "url": "http://example.com"}
```

Returns a dict containing 'package':the updated dataset with fields filtered by include parameter

Return type dictionary

ckan.logic.action.update.**package_resource_reorder**(*context*, *data_dict*)

Reorder resources against datasets. If only partial resource ids are supplied then these are assumed to be first and the other resources will stay in their original order

Parameters

- **id** (*string*) – the id or name of the package to update
- **order** (*list*) – a list of resource ids in the order needed

ckan.logic.action.update.**package_relationship_update**(*context*, *data_dict*)

Update a relationship between two datasets (packages).

The subject, object and type parameters are required to identify the relationship. Only the comment can be updated.

You must be authorized to edit both the subject and the object datasets.

Parameters

- **subject** (*string*) – the name or id of the dataset that is the subject of the relationship
- **object** (*string*) – the name or id of the dataset that is the object of the relationship

- **type** (*string*) – the type of the relationship, one of `'depends_on'`, `'dependency_of'`, `'derives_from'`, `'has_derivation'`, `'links_to'`, `'linked_from'`, `'child_of'` or `'parent_of'`
- **comment** (*string*) – a comment about the relationship (optional)

Returns the updated relationship

Return type dictionary

ckan.logic.action.update.**group_update**(*context*, *data_dict*)
Update a group.

You must be authorized to edit the group.

Note: Update methods may delete parameters not explicitly provided in the data_dict. If you want to edit only a specific attribute use *group_patch* instead.

Plugins may change the parameters of this function depending on the value of the group's `type` attribute, see the *IGroupForm* plugin interface.

For further parameters see *group_create()*.

Parameters **id** (*string*) – the name or id of the group to update

Returns the updated group

Return type dictionary

ckan.logic.action.update.**organization_update**(*context*, *data_dict*)
Update a organization.

You must be authorized to edit the organization.

Note: Update methods may delete parameters not explicitly provided in the data_dict. If you want to edit only a specific attribute use *organization_patch* instead.

For further parameters see *organization_create()*.

Parameters

- **id** (*string*) – the name or id of the organization to update
- **packages** – ignored. use *package_owner_org_update()* to change package ownership

Returns the updated organization

Return type dictionary

ckan.logic.action.update.**user_update**(*context*, *data_dict*)
Update a user account.

Normal users can only update their own user accounts. Sysadmins can update any user account. Can not modify exisiting user's name.

Note: Update methods may delete parameters not explicitly provided in the data_dict. If you want to edit only a specific attribute use *user_patch* instead.

For further parameters see *user_create()*.

Parameters **id** (*string*) – the name or id of the user to update

Returns the updated user account

Return type dictionary

ckan.logic.action.update.**user_generate_apikey**(*context*, *data_dict*)
Cycle a user's API key

Parameters **id** (*string*) – the name or id of the user whose key needs to be updated

Returns the updated user

Return type dictionary

ckan.logic.action.update.**task_status_update**(*context*, *data_dict*)
Update a task status.

Parameters

- **id** (*string*) – the id of the task status to update
- **entity_id** (*string*) –
- **entity_type** (*string*) –
- **task_type** (*string*) –
- **key** (*string*) –
- **value** – (optional)
- **state** – (optional)
- **last_updated** – (optional)
- **error** – (optional)

Returns the updated task status

Return type dictionary

ckan.logic.action.update.**task_status_update_many**(*context*, *data_dict*)
Update many task statuses at once.

Parameters **data** (*list of dictionaries*) – the task_status dictionaries to update, for the format of task status dictionaries see *task_status_update()*

Returns the updated task statuses

Return type list of dictionaries

ckan.logic.action.update.**term_translation_update**(*context*, *data_dict*)
Create or update a term translation.

You must be a sysadmin to create or update term translations.

Parameters

- **term** (*string*) – the term to be translated, in the original language, e.g. `'romantic novel'`
- **term_translation** (*string*) – the translation of the term, e.g. `'Liebesroman'`
- **lang_code** (*string*) – the language code of the translation, e.g. `'de'`

Returns the newly created or updated term translation

Return type dictionary

ckan.logic.action.update.**term_translation_update_many**(*context*, *data_dict*)
Create or update many term translations at once.

Parameters **data** (*list of dictionaries*) – the term translation dictionaries to create or update, for the format of term translation dictionaries see *term_translation_update()*

Returns a dictionary with key `'success'` whose value is a string stating how many term translations were updated

Return type string

ckan.logic.action.update.**vocabulary_update**(*context*, *data_dict*)
Update a tag vocabulary.

You must be a sysadmin to update vocabularies.

For further parameters see *vocabulary_create()*.

Parameters **id** (*string*) – the id of the vocabulary to update

Returns the updated vocabulary

Return type dictionary

ckan.logic.action.update.**dashboard_mark_activities_old**(*context*, *data_dict*)
Mark all the authorized user's new dashboard activities as old.

This will reset *dashboard_new_activities_count()* to 0.

ckan.logic.action.update.**send_email_notifications**(*context*, *data_dict*)
Send any pending activity stream notification emails to users.

You must provide a sysadmin's API key in the Authorization header of the request, or call this action from the command-line via a *paster post ...* command.

ckan.logic.action.update.**package_owner_org_update**(*context*, *data_dict*)
Update the owning organization of a dataset

Parameters

- **id** (*string*) – the name or id of the dataset to update
- **organization_id** (*string*) – the name or id of the owning organization

ckan.logic.action.update.**bulk_update_private**(*context*, *data_dict*)
Make a list of datasets private

Parameters

- **datasets** (*list of strings*) – list of ids of the datasets to update
- **org_id** (*int*) – id of the owning organization

ckan.logic.action.update.**bulk_update_public**(*context*, *data_dict*)
Make a list of datasets public

Parameters

- **datasets** (*list of strings*) – list of ids of the datasets to update
- **org_id** (*int*) – id of the owning organization

ckan.logic.action.update.**bulk_update_delete**(*context*, *data_dict*)
Make a list of datasets deleted

Parameters

- **datasets** (*list of strings*) – list of ids of the datasets to update

- **org_id** (*int*) – id of the owning organization

ckan.logic.action.update.**config_option_update**(*context*, *data_dict*)

New in version 2.4.

Allows to modify some CKAN runtime-editable config options

It takes arbitrary key, value pairs and checks the keys against the config options update schema. If some of the provided keys are not present in the schema a `ValidationError` is raised. The values are then validated against the schema, and if validation is passed, for each key, value config option:

- It is stored on the `system_info` database table
- The Pylons `config` object is updated.
- The `app_globals` (`g`) object is updated (this only happens for options explicitly defined in the `app_globals` module.

The following lists a `key` parameter, but this should be replaced by whichever config options want to be updated, eg:

```
get_action('config_option_update)({}, {
    'ckan.site_title': 'My Open Data site',
    'ckan.homepage_layout': 2,
})
```

Parameters **key** (*string*) – a configuration option key (eg `ckan.site_title`). It must be present on the `update_configuration_schema`

Returns a dictionary with the options set

Return type dictionary

Note: You can see all available runtime-editable configuration options calling the *config_option_list()* action

---

Note: Extensions can modify which configuration options are runtime-editable. For details, check *Making configuration options runtime-editable*.

Warning: You should only add config options that you are comfortable they can be edited during runtime, such as ones you've added in your own extension, or have reviewed the use of in core CKAN.

4.9.4 ckan.logic.action.patch

New in version 2.3. API functions for partial updates of existing data in CKAN

ckan.logic.action.patch.**package_patch**(*context*, *data_dict*)

Patch a dataset (package).

Parameters **id** (*string*) – the id or name of the dataset

The difference between the update and patch methods is that the patch will perform an update of the provided parameters, while leaving all other parameters unchanged, whereas the update methods deletes all parameters not explicitly provided in the data_dict.

You are able to partially update and/or create resources with package_patch. If you are updating existing resources be sure to provide the resource id. Existing resources excluded from the package_patch data_dict will be removed. Resources in the package data_dict without an id will be treated as new resources and will be added. New resources added with the patch method do not create the default views.

You must be authorized to edit the dataset and the groups that it belongs to.

ckan.logic.action.patch.**resource_patch**(*context*, *data_dict*)
Patch a resource

Parameters **id** (*string*) – the id of the resource

The difference between the update and patch methods is that the patch will perform an update of the provided parameters, while leaving all other parameters unchanged, whereas the update methods deletes all parameters not explicitly provided in the data_dict

ckan.logic.action.patch.**group_patch**(*context*, *data_dict*)
Patch a group

Parameters **id** (*string*) – the id or name of the group

The difference between the update and patch methods is that the patch will perform an update of the provided parameters, while leaving all other parameters unchanged, whereas the update methods deletes all parameters not explicitly provided in the data_dict

ckan.logic.action.patch.**organization_patch**(*context*, *data_dict*)
Patch an organization

Parameters **id** (*string*) – the id or name of the organization

The difference between the update and patch methods is that the patch will perform an update of the provided parameters, while leaving all other parameters unchanged, whereas the update methods deletes all parameters not explicitly provided in the data_dict

4.9.5 ckan.logic.action.delete

API functions for deleting data from CKAN.

ckan.logic.action.delete.**user_delete**(*context*, *data_dict*)
Delete a user.

Only sysadmins can delete users.

Parameters **id** (*string*) – the id or usernamename of the user to delete

ckan.logic.action.delete.**package_delete**(*context*, *data_dict*)
Delete a dataset (package).

This makes the dataset disappear from all web & API views, apart from the trash.

You must be authorized to delete the dataset.

Parameters **id** (*string*) – the id or name of the dataset to delete

ckan.logic.action.delete.**dataset_purge**(*context*, *data_dict*)
Purge a dataset.

> **Warning:** Purging a dataset cannot be undone!

Purging a database completely removes the dataset from the CKAN database, whereas deleting a dataset simply marks the dataset as deleted (it will no longer show up in the front-end, but is still in the db).

You must be authorized to purge the dataset.

Parameters **id** (*string*) – the name or id of the dataset to be purged

ckan.logic.action.delete.**resource_delete**(*context*, *data_dict*)
Delete a resource from a dataset.

You must be a sysadmin or the owner of the resource to delete it.

Parameters **id** (*string*) – the id of the resource

ckan.logic.action.delete.**resource_view_delete**(*context*, *data_dict*)
Delete a resource_view.

Parameters **id** (*string*) – the id of the resource_view

ckan.logic.action.delete.**resource_view_clear**(*context*, *data_dict*)
Delete all resource views, or all of a particular type.

Parameters **view_types** (*list*) – specific types to delete (optional)

ckan.logic.action.delete.**package_relationship_delete**(*context*, *data_dict*)
Delete a dataset (package) relationship.

You must be authorised to delete dataset relationships, and to edit both the subject and the object datasets.

Parameters

- **subject** (*string*) – the id or name of the dataset that is the subject of the relationship
- **object** (*string*) – the id or name of the dataset that is the object of the relationship
- **type** (*string*) – the type of the relationship

ckan.logic.action.delete.**member_delete**(*context*, *data_dict=None*)
Remove an object (e.g. a user, dataset or group) from a group.

You must be authorized to edit a group to remove objects from it.

Parameters

- **id** (*string*) – the id of the group
- **object** (*string*) – the id or name of the object to be removed
- **object_type** (*string*) – the type of the object to be removed, e.g. `package` or `user`

ckan.logic.action.delete.**package_collaborator_delete**(*context*, *data_dict*)
Remove a collaborator from a dataset.

Currently you must be an Admin on the dataset owner organization to manage collaborators.

Note: This action requires the collaborators feature to be enabled with the *ckan.auth.allow_dataset_collaborators* configuration option.

Parameters

- **id** (*string*) – the id or name of the dataset
- **user_id** (*string*) – the id or name of the user to remove

ckan.logic.action.delete.**group_delete**(*context*, *data_dict*)
Delete a group.

You must be authorized to delete the group.

Parameters **id** (*string*) – the name or id of the group

ckan.logic.action.delete.**organization_delete**(*context*, *data_dict*)
Delete an organization.

You must be authorized to delete the organization and no datasets should belong to the organization unless 'ckan.auth.create_unowned_dataset=True'

Parameters **id** (*string*) – the name or id of the organization

ckan.logic.action.delete.**group_purge**(*context*, *data_dict*)
Purge a group.

Warning: Purging a group cannot be undone!

Purging a group completely removes the group from the CKAN database, whereas deleting a group simply marks the group as deleted (it will no longer show up in the frontend, but is still in the db).

Datasets in the organization will remain, just not in the purged group.

You must be authorized to purge the group.

Parameters **id** (*string*) – the name or id of the group to be purged

ckan.logic.action.delete.**organization_purge**(*context*, *data_dict*)
Purge an organization.

Warning: Purging an organization cannot be undone!

Purging an organization completely removes the organization from the CKAN database, whereas deleting an organization simply marks the organization as deleted (it will no longer show up in the frontend, but is still in the db).

Datasets owned by the organization will remain, just not in an organization any more.

You must be authorized to purge the organization.

Parameters **id** (*string*) – the name or id of the organization to be purged

ckan.logic.action.delete.**task_status_delete**(*context*, *data_dict*)
Delete a task status.

You must be a sysadmin to delete task statuses.

Parameters **id** (*string*) – the id of the task status to delete

ckan.logic.action.delete.**vocabulary_delete**(*context*, *data_dict*)
Delete a tag vocabulary.

You must be a sysadmin to delete vocabularies.

Parameters **id** (*string*) – the id of the vocabulary

ckan.logic.action.delete.**tag_delete**(*context*, *data_dict*)
Delete a tag.

You must be a sysadmin to delete tags.

Parameters

- **id** (*string*) – the id or name of the tag
- **vocabulary_id** (*string*) – the id or name of the vocabulary that the tag belongs to (optional, default: None)

ckan.logic.action.delete.**unfollow_user**(*context*, *data_dict*)
Stop following a user.

Parameters **id** (*string*) – the id or name of the user to stop following

ckan.logic.action.delete.**unfollow_dataset**(*context*, *data_dict*)
Stop following a dataset.

Parameters **id** (*string*) – the id or name of the dataset to stop following

ckan.logic.action.delete.**group_member_delete**(*context*, *data_dict=None*)
Remove a user from a group.

You must be authorized to edit the group.

Parameters

- **id** (*string*) – the id or name of the group
- **username** (*string*) – name or id of the user to be removed

ckan.logic.action.delete.**organization_member_delete**(*context*, *data_dict=None*)
Remove a user from an organization.

You must be authorized to edit the organization.

Parameters

- **id** (*string*) – the id or name of the organization
- **username** (*string*) – name or id of the user to be removed

ckan.logic.action.delete.**unfollow_group**(*context*, *data_dict*)
Stop following a group.

Parameters **id** (*string*) – the id or name of the group to stop following

ckan.logic.action.delete.**job_clear**(*context*, *data_dict*)
Clear background job queues.

Does not affect jobs that are already being processed.

Parameters **queues** (*list*) – The queues to clear. If not given then ALL queues are cleared.

Returns The cleared queues.

Return type list

New in version 2.7.

ckan.logic.action.delete.**job_cancel**(*context*, *data_dict*)
Cancel a queued background job.

Removes the job from the queue and deletes it.

Parameters **id** (*string*) – The ID of the background job.

New in version 2.7.

ckan.logic.action.delete.**api_token_revoke**(*context*, *data_dict*)
Delete API Token.

Parameters

- **token** (*string*) – Token to remove(required if *jti* not specified).
- **jti** (*string*) – Id of the token to remove(overrides *token* if specified).

New in version 3.0.