HDX CKAN API Cookbook

Accessing data through the CKAN API

The Humanitarian Data Exchange (HDX) runs on a platform called CKAN. You can use CKAN's API to discover, update, and download data from HDX without human intervention.

This cookbook focuses on data access rather than data updating (if you're interested in automated data updating, we have a Python library hdx-python-api that will make the process much simpler).

We'll start with some basic ingredients, then move on to some full recipes for data access. In each case, we'll show both the direct RESTful API URL, and the Python code that you can use via the official ckanapi (which also includes command-line utilities for use in shell scripts).

For a simpler method to get automatic notifications about new or modified datasets matching any search criteria (e.g. for a specific organization, country, tag, search string, or combination of those), see Appendix A. Syndication feeds for notifications.

1. Basics

This section introduces the fundamental information you'll need to know to connect to the API. You may choose to skip straight to 3. Simple search examples if you prefer to see some examples first.

1.1. Connecting to the API

The root URL for the CKAN API in HDX is

https://data.humdata.org/api/3/

To use this with the CKAN Python API library, try

```
from ckanapi import RemoteCKAN
ckan = RemoteCKAN("https://data.humdata.org")
```

(Note that in Python, you supply just the CKAN domain name, not the full API path.) For all the Python examples that follow, we'll assume that you've created this ckan object already.

1.2. Reading a single dataset with package_show

A useful API call is retrieving the JSON representation of a dataset's metadata. On HDX we refer to "datasets", the CKAN API uses the term "package" for "dataset". This example grabs the dataset for Humanitarian Response Plan projects for Nigeria:

https://data.humdata.org/api/3/action/package_show?id=hrp-projects-nga

or in Python:

```
package =
ckan.action.package_show(id="hrp-projects-nga")
```

(In Python, you'll get the JSON result directly; with the URL, you'll get the metadata response under a key called "result".) The string "hrp-projects-nga", is the same stub that appears at the end of the dataset URL on HDX, https://data.humdata.org/dataset/hrp-projects-nga.

Exercise: try using the patterns above to read the metadata for other datasets on ckan.

1.3. Anatomy of a dataset

When you try the queries above, you'll note that the JSON does not contain the actual data. To get at that, it's necessary to understand the basic structure of the metadata returned (there are many other properties, but we'll stick with these for now):

```
{
    "name": "...",
    "title": "...",
    "description": "...",
    "created": "...",
    "last_modified": "...",
    "dataset_date": "...",
    "dataseries_name": "...",
    "groups": [
        { … },
        { … }
    ],
    "organization": { … },
    "resources": [
        { … },
        { … }
    ],
    "tags" [
        { … },
        { … }
    ]
}
```

name: the dataset stub on HDX, like "hrp-projects-nga"

title: the full human-readable dataset title, like "Humanitarian Response Plan projects for Nigeria"

description: a longer description of the dataset.

created: the date and time when the dataset was first created on ckan.

last_modified: the date and time when the dataset *metadata* on HDX was last changed (not the data itself can change independently, especially if it's hosted off HDX).

dataset_date: the date range when the data is applicable (could be in the past or future relative to the creation and last-modified dates).

dataseries_name: (optional) a curated series, or list, to which this dataset belongs.

groups: a list of data structures describing countries or country-like entities associated with the dataset (in this case, just Nigeria).

organization: a data structure describing the data provider (e.g. OCHA Financial Tracking Service).

resources: a list of data structures describing the resources (files) inside the dataset.

tags: a list of data structures describing the semantic tags associated with the dataset (like "who is doing what and where-3w-4w-5w").

(You can learn more about the different data structures and properties at https://docs.ckan.org/en/api)

1.3.1. Downloading the data

To get at the actual data, you need to go inside the resources list, pick a resource (the first one is often a good choice), and use the download_url property to download the actual dataset. A download_url will look something like this (though it could also point directly to site outside HDX):

> https://data.humdata.org/dataset/3527869c-8fe9-4289-9d57-1811e789bf60/resource/96b24403-0de4-4652-bb76-f585c04b5e6d/download/admin1-summaries-litpop.csv

We'll talk more about using the different parts of the dataset metadata later. The main focus of this cookbook will be how to locate datasets on HDX so that your code can download and process them with minimal human intervention.

2. Searching HDX with package_search

If you are the kind of person who likes to start with basic principles, then go ahead and read this section now. If you prefer to learn by example, then feel free to skip ahead to 3. Simple search examples, and then come back here later to fill in any gaps.

To find datasets, we'll use the *package_search* endpoint at

> https://data.humdata.org/api/3/action/package_search

or, in Python

```
packages = ckan.action.package_search()
```

(Calling this without parameters returns all the datasets on HDX.)

2.1. Paging through results

CKAN search results are paged: the start parameter gives the starting position(zero-based), while the rows parameter gives the page size. To get all of the results, choose a page size (e.g. 100), then keep advancing start by that amount in each query until you have all of the results. For example, the following query gives the third page of 100 public packages/datasets on HDX:

https://data.humdata.org/api/3/action/package_search?start=200&rows=100

or, in Python

```
packages = ckan.action.package_search(start=200,
rows=100)
```

Tip: instead of doing your own paging, you can use the ckancrawler Python package, which feeds search results smoothly into a single iterator like this, doing all the paging behind the scenes:

```
from ckancrawler import Crawler
crawler = Crawler("https://data.humdata.org")
for package in crawler.packages():
  // do something with the package
```

2.2. Constructing queries

For this cookbook, we use the *q* parameter to queries, which corresponds to the search text used at https://data.humdata.org/search

(There is an alternative parameter named *fq* that works the same way but doesn't affect search-result weighting for relevancy. You can ignore it for the sake of the examples in this cookbook, but you might find it useful if you decide to try more-complex free-text searches in the future.)

If you're constructing a URL query directly, then you will have to URL-encode your search string, so that "displaced people" (for example) becomes "displaced%20people" or "displaced+people":

> https://data.humdata.org/api/3/action/package_search?q=displaced%20people

In Python, the *ckanapi* will do the escaping for you:

```
packages = ckan.action.package_search(q="displaced people")
```

2.2.1. Query filters

Queries can address specific metadata fields using a syntax like ***fieldname*:**query term:

groups: datasets related to this country or country-like entity, using the HDX country stub, which is usually the ISO3 code in lower case (more information).

organization: datasets from this provider, using the HDX organization stub (more information).

vocab_Topics: datasets labelled with this thematic tag (more information).

dataseries_name: datasets belonging to this data series (more information).

For example, this URL retrieves datasets provided by FAO (440 of them as of November 2024), using the HDX organization stub "fao":

> https://data.humdata.org/api/3/action/package_search?q=organization:fao

Python code:

```
packages = ckan.action.package_search(q="organization:fao")
```

For a complete list of query filters available for HDX, see B.1. Complete list of HDX CKAN search fields.

2.2.2. Advanced boolean logic

By default, repeated filters imply boolean AND: if you have a query "`groups:afg group:pak`" it will include only datasets that apply to Afghanistan *and* to Pakistan. For advanced use (beyond what's needed for the examples in this cookbook), you can use special Solr filter syntax. For example, the query "`group:afg OR group:pak`" will return datasets associated with either or both countries:

> https://data.humdata.org/api/3/action/package_search?q=groups:afg%20OR%20groups:pak

Python code:

```
packages = ckan.action.package_search(
  q="groups:afg OR groups:pak"
)
```

2.3. Sorting results

The optional *sort* parameter allows you to control the order in which you get your search results. After each of the following, you can add "asc" (ascending order; the default) or "desc" (descending order):

metadata_created: when the dataset was first created in ckan.

last_modified: the last time the dataset was changed on HDX (will not detect changes to remote resources, like APIs).

score: relevance to your search query (add *desc* to get the most-relevant datasets first).

title_case_insensitive: title of the dataset.

pageviews_last_14_days: trending (add *desc* to get the most-popular ones first).

total_res_downloads: number of data downloads (add *desc* to get the most-downloaded ones first).

You can combine these if you wish.

The following URL returns datasets sorted by total downloads in descending order (so that the most-popular ones appear first); note that the whitespace needs to be URL-encoded:

https://data.humdata.org/api/3/action/package_search?sort=total_res_downloads%20desc

Python code:

```
packages = ckan.action.package_search(
  sort="total_res_download desc"
)
```

2.4. Format of search results

Search results using a raw URL look like this:

```
{
  "help": "…",
  "success": true,
  "result": {
    count: …,
    facets: { … },
    expanded: { … },
    results: [
      { … },
      { … }
    ],
  }
}
```

The Python API strips off the top layer, so you see just the following:

```
{
  count: …,
  facets: { … },
  expanded: { … },
  results: [
    { … },
    { … }
  ],
}
```

Only two of these fields are essential:

count: the total results available (not just the ones returned from this paged query)

results: a list of packages/dataset metadata objects, as described in 1.3. Anatomy of a dataset.

3. Simple search examples

This section contains simple examples to illustrate the search principles introduced in 2. Searching HDX with package_search. Note that we are searching only dataset *metadata*, not the data itself. More-advanced examples appear in 4. Complex queries.

3.1. Finding datasets by country/group

As mentioned in 2.2.1. Query filters, HDX represents countries (and country-like entities) using the CKAN *groups* parameter. HDX group stubs are mostly identical to ISO3 country codes, in lower case, so the code for Ukraine is "ukr" (*not* "UKR" or "UA"). A full list of HDX countries/groups is available from

> https://data.humdata.org/api/3/action/group_list?all_fields=true

or in Python,

```
countries = ckan.action.group_list(all_fields=True)
```

Use the name field in your query (that contains the code).

3.1.1. Example: Ukraine

The following query will return public datasets for Ukraine (215 of them as of November 2024):

> https://data.humdata.org/api/3/action/package_search?q=groups:ukr

or in Python,

```
packages = ckan.action.package_search(q="groups:ukr")
```

As described in 2.3. Sorting results, if you want the most-recent datasets first, you can add a sort parameter:

https://data.humdata.org/api/3/action/package_search?q=groups:ukr&sort=metadata_created%20desc

or in Python,

```
packages = ckan.action.package_search(
  q="groups:ukr",
  sort="metadata_created desc"
)
```

3.2. Finding datasets by provider

Data providers in HDX use the organization parameter, introduced in 2.2.1. Query filters. A complete list of HDX data-provider organizations is available at

https://data.humdata.org/api/3/action/organization_list?all_fields=true

or in Python,

```
orgs = ckan.action.organization_list(all_fields=True)
```

As with countries, use the name field from these results in your queries.

3.2.1. Example: Integrated Food Security Phase Classification

Looking in the link above, the Integrated Food Security Phase Classification (IPC) has the HDX stub (name) "ipc", so you should use the identifier "ipc" to find datasets provided by the organisation. Alternatively, you can find the same stub at the end of the URL organization page https://data.humdata.org/organization/ipc

The following query will return a list of IPC's datasets (57 of them as of November 2024):

https://data.humdata.org/api/3/action/package_search?fq=organization:ipc

or in Python,

```
packages =
ckan.action.package_search(q="organization:ipc")
```

3.3. Finding datasets by topic tag

HDX uses a special controlled CKAN tag vocabulary named "Topics" for thematic tagging of datasets. In searches, you query a topic using the *vocab_Topics* parameter introduced in 2.2.1. Query filters.

A list of topic tags is available at

https://data.humdata.org/api/3/action/tag_list?vocabulary_id=Topics&all_fields=true

or in Python,

```
topics = ckan.action.tag_list(
  vocabulary_id="Topics",
  all_fields=True
)
```

As with countries and organizations, use the name field from these results in your queries.

3.3.1. Example: Gender-based violence

The topic tag "gender-based violence-gbv" will allow us to find HDX datasets related to gender-based violence:

https://data.humdata.org/api/3/action/package_search?q=vocab_Topics:%22gender-based%20violence-gbv%22

WARNING: If the topic tag contains whitespace (as many in HDX unfortunately do), you will have to both URL-encode the whitespace *and* surround the tag name in quotation marks when constructing the URL. This is an easy trap to fall into when working with the HDX CKAN API.

In Python, the quotation marks are also required (but not, obviously, the URL encoding):

```
packages = ckan.action.package_search(
  q="vocab_Topics:\"gender-based violence-gbv\""
)
```

3.4. Finding datasets by data series

HDX data series group datasets by source and theme, for example "IOM - DTM Baseline Assessment". These are an effective way to find datasets automatically when they're created. In searches, you query data series using the *dataseries_name* parameter introduced in 2.2.1. Query filters.

A list of data series is available at

https://data.humdata.org/api/3/action/package_search?rows=0&fq=dataset_type:dataset&facet.field=[%22dataseries_name%22]&facet.limit=1000

Use the *dataseries_name* field from these results in your queries.

3.4.1. Example: IOM DTM Baseline Assessments

As mentioned above, the data-series name "IOM - DTM Baseline Assessment" will allow you to find HDX datasets containing IOM's DTM baseline assessments for various countries (remember to quote the name and URL-encode any whitespace in it):

https://data.humdata.org/api/3/action/package_search?q=dataseries_name:%22IOM%20-%20DTM%20Baseline%20Assessment%22

WARNING: If the data-series name contains whitespace (as many in HDX unfortunately do), you will have to both URL-encode the whitespace *and* surround the tag name in quotation marks when constructing the URL. This is an easy trap to fall into when working with the HDX CKAN API.

In Python, you don't need to URL-encode the whitespace, but you still need to quote it:

```
packages = ckan.action.package_search(
```

```
    q="dataseries_name:\"IOM - DTM Baseline
Assessment\""
)
```

3.5. Finding datasets by free-text search

In addition to the special fields described in 2.2.1. Query filters, you can choose to simply search for a text string that's likely to appear in a dataset's title or description. For example, the following will find all datasets that mention "volunteers" (10 of them as in December 2024):

> https://data.humdata.org/api/3/action/package_search?q=volunteers

In Python, you would use

```
packages = ckan.action.package_search(q="volunteers")
```

For information on using wildcards in text searches, see B.3. Querying with wildcards and ranges.

4. Complex queries

The API calls in 3. Simple search examples won't always be enough. To locate a specific dataset with confidence, you will need to combine multiple filters, sorting specifications, and (in some cases) free-text search. The following examples show how your code can use these together to find a relevant dataset automatically.

4.1. Latest OCHA 3W for Lebanon

Activity reports ("Who? What? Where?," or simply "3W") are a core humanitarian data type. Many international organisations and humanitarian clusters produce their own 3Ws, but for this example, we want to get the latest consolidated 3W, a cross-sector dataset which UNOCHA coordinates in countries where it works.

As of December 2024, there is no data series for all OCHA 3Ws, but there is a general CKAN topic tag "`who is doing what and where-3w-4w-5w`" (see 3.3. Finding datasets by topic tag), so this makes a good starting point:

```
vocab_Topics:"who is doing what and where-3w-4w-5w"
```

Next, we want to narrow the results down to Lebanon (see 3.1. Finding datasets by country/group), so we add the filter

```
group:leb
```

And we want only 3Ws from OCHA, not from other organisations (see 3.2. Finding datasets by provider). That's trickier, because each OCHA field office is a separate organisation on ckan. For OCHA Lebanon, the HDX organisation is "`ocha-lebanon`". So we also add

```
organization:ocha-lebanon
```

(In this case, we could have left out the country filter, but in others, one OCHA field office might produce 3Ws for multiple countries, so it's usually best to leave it in.)

And finally, we'll want to sort the results so that the latest 3W appears first (see 2.3. Sorting results), so we will add the sort parameter

```
last_modified desc
```

When we put it all together, we end up with this API call:

```
https://data.humdata.org/api/3/action/package_search?q=vocab_Topics%3A%22who%20is%20doing%20what%20and%20where-3w-4w-5w%22%20groups:lbn%20organization:ocha-lebanon&sort=last_modified%20desc
```

or in Python,

```
query_parts = (
  "vocab_Topics:\"who is doing what and
where-3w-4w-5w\"",
  "groups:lbn",
  "organization:ocha-lebanon",
)

packages = ckan.action.package_search(
  q=" ".join(query_parts),
  sort="last_modified desc"
)
```

The first result should reliably be the latest available OCHA 3W for Lebanon. This will work even if the dataset name and URL change on HDX.

4.2. Food prices for Venezuela

For this exercise, we want to find the latest food prices for Venezuela. We can find the data series "`WFP - Food Prices`" using the method described in 3.4. Finding datasets by data series, which gets us a good part of the way there.

```
dataseries_name:"WFP - Food Prices"
```

We also need to set the group to "ven" for Venezuela (see 3.1. Finding datasets by country/group):

```
groups:ven
```

And finally, once again, we want to set the sort to

```
last_modified desc
```

so that the first result will be the most-recent one in case there is more than one result (see 2.3. Sorting results).

All together, that gives us the following API call:

```
https://data.humdata.org/api/3/action/package_search?q=dataseries_name:%22WFP%20-%20Food%20Prices%22%20groups:ven&sort=last_modified%20desc
```

In Python, we can retrieve the same data like this:

```
query_parts = (
  "dataseries_name:\"WFP - Food Prices\"",
  "groups:ven",
)
packages = ckan.action.package_search(
  q=" ".join(query_parts),
  sort="last_modified desc"
)
```

4.4. Sex- and age-disaggregated datasets related to refugees

Now, let's try a more-thematic approach. We want to find all datasets that contain sex- and age-disaggregated data about refugees. In this case, we want to combine two topic tags, as introduced in 3.3. Finding datasets by topic tag (note that we have to quote the first one because of the internal spaces):

```
vocab_Topics:"sex and age disaggregated data-sadd"
vocab_Topics:refugees
```

These result in the following API call:

https://data.humdata.org/api/3/action/package_search?q=vocab_Topics:%22sex%20and%20age%20disaggregated%20data-sadd%22%20vocab_Topics:refugees

In Python, you can use the following code:

```
query_parts = (
  "vocab_Topics:\"sex and age disaggregated
data-sadd\"",
  "vocab_Topics:refugees",
)
packages = ckan.action.package_search(
  q=" ".join(query_parts)
)
```

Appendix A. Syndication feeds for notifications

As an alternative to using the CKAN API to find data on HDX, you can use Atom (similar to RSS) syndication feeds to receive notifications of any new or modified datasets matching your search criteria. You construct searches the same way as for the CKAN API, but the URL pattern looks like this (using a simple text search for "food"):

https://data.humdata.org/feeds/dataset.atom?q=food

The result is a list of entries like this (in XML), though libraries for all major programming languages ensure that you will never have to deal with the XML markup directly:

```
<entry>
     <id>https://data.humdata.org/dataset/efad2587-3c06-
4530-ba12-1c6e8ae393db</id>
     <title>Guinea - HungerMap data</title>
     <updated>2024-12-10T14:13:31.694584+00:00</updated>
     <content>HungerMapLIVE is ...</content>
     <link
href="https://data.humdata.org/api/3/action/package_show
?id=efad2587-3c06-4530-ba12-1c6e8ae393db"
rel="alternate"/>
     <link
href="https://data.humdata.org/api/3/action/package_show
?id=wfp-hungermap-data-for-gin" rel="enclosure"/>
     <category term="food security"/>
     <category term="hxl"/>
     <category term="indicators"/>

<published>2024-11-25T21:44:13.878433+00:00</published>
  </entry>
```

Note that the entry contains a direct link into the HDX CKAN API to download the package metadata (see 1.3. Anatomy of a dataset). The first entry will be the most-recently-modified result, and so on, so you can use this to get automated update notifications for any of the searches described in the cookbook.

There are also simpler, dedicated URLs to get updates for a country or organization without using the fielded search syntax. For example, this feed will always return the most-recently-updated public datasets related to Afghanistan:

https://data.humdata.org/feeds/group/afg.atom

And this feed will always return the latest public datasets provided by the World Food Programme:

https://data.humdata.org/feeds/organization/wfp.atom

You can consume these feeds programmatically using a library like atoma in Python, or you can load them into a feed reader like Feedly or NetNewsWire for human consumption (beside blogs, news articles, and other syndicated information).

Appendix B: Advanced CKAN search features

(Contributed by Ian Hopkinson)

CKAN and HDX provide more-advanced search features that aren't covered in this cookbook, but might be useful for specific needs.

B.1. Complete list of HDX CKAN search fields

Here is the complete list of HDX CKAN search fields (as introduced in 2.2.1. Query filters). Those fields prefixed `res_` refer to resource metadata, the rest to dataset metadata. Those fields marked with a * are date type fields which support some special search features described below. The approved tags which appear on HDX datasets are actually stored in a field called "vocab_Topics".

archived	batch	caveats	creator_user_id
data_update_frequency	dataseries_name	dataset_preview	dataset_source
has_geodata	has_quickcharts	has_showcases	id
is_requestdata_type	isopen	last_modified*	license_id
license_title	maintainer	metadata_created*	metadata_modified*
methodology	name	notes	num_of_showcases
num_resources	num_tags	organization	overdue_date*
owner_org	package_creator	pageviews_last_14_days	qa_completed
res_description	res_extras_broken_link	res_extras_in_hapi	res_format
res_name	res_url	review_date*	solr_additions
state	subnational	title	total_res_downloads
type	updated_by_script	url	vocab_Topics

B.2. Querying date fields

CKAN's date search facilities are powerful but not always obvious. You can do an exact search for a datetime with a query like metadata_created:"2019-12-04T10:23:27.806321Z", note that if the trailing Z is omitted the search fails with an Invalid Date String error, however dates are returned from package_search without a trailing Z! Date fields can also be queried with a range expression which allows for the special values NOW, DAY, MONTH, YEAR, HOUR, MINUTE, these can be combined with "normal" dates with +,- and / operators (/ is rounding):

Example:

Find datasets modified in the last 24 hours:

> https://data.humdata.org/api/action/package_search?q=last_modified:[NOW-1DAY%20TO%20NOW]

or in Python,

```
packages = ckan.action.package_search(
  q="last_modified:[NOW-1DAY TO NOW]"
)
```

B.3. Querying with wildcards and ranges

The multicharacter (*) and single character (?) wildcard operators are supported but cannot be used in quoted search terms (use a backslash instead to escape whitespace).

Example:

Find all datasets with the source "ETH Zurich Cli?ada" where "?" represents any letter:

> https://data.humdata.org/api/action/package_search?q=dataset_source:ETH\%20Zurich\%20Cli?ada

or in Python,

```
packages = ckan.action.package_search(
  q="dataset_source:ETH\\ Zurich\\ Cli?ada"
)
```

Example:

Find all datasets with the source "ETH Zurich Cli*" where "*" represents 0 or more letters:

https://data.humdata.org/api/action/package_search?q=dataset_source:ETH\%20Zurich\%20Cli*

or in Python,

```
packages = ckan.action.package_search(
  q="dataset_source:ETH\\ Zurich\\ Cli*"
)
```

As well these simple wildcard usages the search API also supports range queries which can include wildcards.

Example:

Find datasets that have between 2 and 5 resources:

https://data.humdata.org/api/action/package_search?q=num_resources:[2%20TO%205]

Or in Python,

```
packages = ckan.action.package_search(
  q="num_resources:[2 TO 5]"
)
```

Range queries can be used to select values which are not null with the query

```
fieldname:[* TO *]
```

This is the approved way of doing such selections. The query

```
fieldname:*
```

includes datasets that have null values as well as all other values.