

Package: openFDA (via r-universe)

September 16, 2026

Title 'openFDA' API

Version 0.1.0.9002

Description The 'openFDA' API facilitates access to U.S. Food and Drug Administration (FDA) data on drugs, devices, foodstuffs, tobacco, and more. This package makes the API easily accessible with 'httr2', returning objects which the user can convert to JSON data and parse. The API is described in: Kass-Hout TA, Xu Z, Mohebbi M et al. (2016) <[doi:10.1093/jamia/ocv153](https://doi.org/10.1093/jamia/ocv153)>.

License GPL (>= 3)

URL <https://github.com/simpar1471/openFDA>,
<https://simpar1471.github.io/openFDA/>

BugReports <https://github.com/simpar1471/openFDA/issues>

Depends R (>= 4.1.0)

Imports cli, httr2, checkmate, purrr (>= 0.2.3), rlang, vctrs (>= 0.4.0), prettyunits, keyring, memoise, utils

Config/Needs/website rmarkdown

Suggests knitr, rmarkdown, stringr, testthat (>= 3.0.0), vcr (>= 0.6.0)

Config/testthat/edition 3

Config/testthat/start-first options

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

VignetteBuilder knitr

Config/pak/sysreqs libsecret-1-dev libssl-dev

Repository <https://simpar1471.r-universe.dev>

Date/Publication 2025-07-22 16:27:31 UTC

RemoteUrl <https://github.com/simpar1471/openfda>

RemoteRef HEAD

RemoteSha 66c975e59eea8c193452e5f89616656376ae29a6

Contents

format_search_term	2
format_sort_term	3
openFDA	4
openFDA_options	8
set_api_key	10

Index	12
--------------	-----------

format_search_term	<i>Format character vectors into search terms for openFDA API queries</i>
--------------------	---

Description

This function is a helper for constructing search queries. Whilst it handles some of the available formatting for openFDA APIs, it does not recapture all of the search term syntax available to you. To get a full appreciation of the openFDA search syntax, see <https://open.fda.gov/apis/advanced-syntax/>.

Usage

```
format_search_term(search, exact = TRUE, mode = "or")
```

Arguments

search	A character vector of length 1 or more. If scalar and unnamed, it will be assumed that you have already formatted your search string to work with the API. If named, the vector will be collapsed to include your various search terms, separated by AND or OR terms based on the value of operator.
exact	A single-length logical vector. When TRUE (the default), individual search terms will be surrounded with double quotes (""). Set exact to FALSE if your search term contains multiple words to be searched on, e.g. <code>c("openfda.generic_name" = "losartan+candesartan")</code> . This parameter only applies if search is a named character vector.
mode	A single-length character vector, which defines how searches in multiple fields should be combined. By default ("or") they will be combined with an 'OR' operator, but you can make an 'AND' operator be used instead ("and"). This argument is case-sensitive and will throw an error if mode is not one of either "and" or "or". This parameter only applies if search is a named character vector.

Value

A character vector of the S3 class `<AsIS>`, with a formatted search term which can be supplied to `openFDA()`.

Note

This function does not check that you're providing accurate field names or search terms. It is up to you to make sure you've provided correctly spelt fields and search terms.

See Also

- `format_sort_term()` performs similar formatting for the sort component of an openFDA query.
- `I()` generates vectors with the `<AsIs>` S3 class.
- `httr2::req_url()` documents why `I()` is applied to the output of this function.

Examples

```
# Provide a formatted search string and the function will do no formatting
format_search_term("openfda.generic_name:verapamil")

# Provide a named vector and the function will format it for you
format_search_term(c("openfda.generic_name" = "verapamil"))

# If providing multiple elements in your search term, use `exact = FALSE`
# to prevent the function from surrounding the term with double quotes.
format_search_term(c("openfda.generic_name" = "verapamil+amlodipine"),
  exact = FALSE)

# Provide a longer named vector and function will merge these with an OR
# operator
format_search_term(c("openfda.generic_name" = "verapamil",
  "openfda.manufacturer_name" = "glaxo*"))

# Or you can set the `mode` argument to merge your search terms with an AND
# operator
format_search_term(c("openfda.generic_name" = "verapamil",
  "openfda.manufacturer_name" = "glaxo*"),
  mode = "and")
```

format_sort_term

Format character vectors into sort terms for openFDA API queries

Description

This function acts as a helper for constructing a sort term in the openFDA API.

Usage

```
format_sort_term(sort)
```

Arguments

sort A single-length character vector of length 1. If unnamed, it will be assumed that you have already formatted your search string to work with the API. If named, the vector will be collapsed to include your field and sorting choice.

Value

A character vector of the S3 class `<AsIS>`, with a formatted search term which can be supplied to `openFDA()`.

Note

This function does not check that you're providing accurate field names or search terms. It is up to you to make sure you've provided correctly spelt fields and search terms.

See Also

- [format_search_term\(\)](#) performs similar formatting for the search component of an openFDA query.
- [I\(\)](#) generates vectors with the `<AsIs>` S3 class.
- [httr2::req_url\(\)](#) documents why [I\(\)](#) is applied to the output of this function.

Examples

```
# Provide a formatted search string and the function will do no formatting
format_sort_term("openfda.generic_name:asc")

# Provide a named vector and the function will format it for you
format_sort_term(c("openfda.generic_name" = "asc"))

# Errors will be thrown if you supply a bad input
try(format_sort_term("receivedate:no_order"))
try(format_sort_term(c("receivedate" = "ascending")))
```

openFDA

Send requests to the openFDA API

Description

Send requests to the openFDA API

Usage

```

openFDA(
  search = "",
  sort = NULL,
  count = NULL,
  limit = 1000,
  skip = NULL,
  endpoint = "drug-drugsfda",
  api_key = get_api_key(),
  paging = current_openFDA_options()[["openFDA.paging"]],
  paging_verbosity = current_openFDA_options()[["openFDA.paging_verbosity"]],
  handle_http_errors = current_openFDA_options()[["openFDA.handle_http_errors"]]
)

```

Arguments

search	A character vector which will be passed to <code>format_search_term()</code> . If search is "" (the default), openFDA will retrieve all records with no filtering. An error will be thrown if any elements of search are missing (NA).
sort	A single string or scalar named character vector describing how to sort the results. The sort term should either be formatted as "[FIELD]:[asc/desc]", or c("[FIELD]" = "[asc/desc]"). For example, to sort results in the Drugs@FDA endpoint by ascending submission status dates. If values other than "asc", or "desc" are supplied, the function will throw an error.
count	A single string denoting a field on which to count results. If NULL (the default), results will be returned in full to the user. Specify this parameter if you want to count occurrences of results within your search term. The openFDA website has more information on how to count by a specific field.
limit	A single integerish value describing the limit on the number of records to retrieve. An error will be thrown if limit is more than 1000 (the default).
skip	A single integerish value describing how many records should be skipped. If more records are skipped than are found in your search, the openFDA API will return a 404 error.
endpoint	A single-length character vector describing which openFDA endpoint to target (case-sensitive). By default, openFDA() targets the Drugs@FDA endpoint ("drug-drugsfda"). • "animalandveterinary-event": Adverse event reports for animal/veterinary drugs. • "drug-event": Adverse event reports from FDA Adverse Event Reporting System • "drug-label": Drug documentation in the Structured Product Labeling (SPL) format. • "drug-ndc": Data from the National Drug Code Directory (NDC). • "drug-enforcement": Data from the FDA Recall Enterprise System about drug products.

- "drug-drugsfda": Data on products approved for human use by the FDA since 1939, with mostly complete data after 1998.
- "drug-shortages": Data on [drug shortages](#) as compiled by the FDA.
- "device-510k": Data from 510(k) submissions to the FDA regarding medical devices.
- "device-classification": Data from the FDA [Product Classification Database](#) for medical devices.
- "device-enforcement": Data from the [FDA Recall Enterprise System \(RES\)](#) for medical devices.
- "device-event": Data on adverse events associated with FDA-approved medical devices.
- "device-pma": Pre-market approval data for medical devices submitted to the FDA.
- "device-recall": Data on product recalls for medical devices which violate FDA law.
- "device-registrationlisting": Data on FDA [Device Registrations and Listings](#).
- "device-covid19serology": Data from [independent evaluations of COVID-19 serological tests](#).
- "device-udi": Data from the FDA's Global Unique Device Identification Database (GUDID).
- "food-enforcement": Data from the [FDA Recall Enterprise System](#) for food products.
- "food-event": Data from the [Center for Food Safety and Applied Nutrition Adverse Event Reporting System](#).
- "other-historicaldocument": The majority of FDA Press Releases, from 1913 to 2014.
- "other-nsde": Data from the [National Drug Code Structured Product Labeling Data Elements](#).
- "other-substance": Data regarding substances - individual molecules, proteins, nucleic acids, and more.
- "other-unii": Data from the [FDA's Global Substance Registration System](#).
- "tobacco-problem": Data on problems (e.g. damage, defects, contamination, bad smell) with tobacco products.

`api_key` A single-length character vector with your openFDA API key. By default this is the result of `get_api_key()`. You must set this with `set_api_key()`, or `openFDA()` will not work.

`paging` A single string describing whether results should be paged. `openFDA()` uses the value of the package-level option `openFDA.paging` for this argument, but you can set this behaviour on a per-function call level with this parameter. You can edit the option itself with `openFDA_options()`. Permissible values for `paging_verbosity` include:

- "ask" - `openFDA()` will warn you that pagination is required and ask if you want this to be done. Depending on user input, either a single [http2](#) response

object or a list of **httr2** response objects will be returned. This will throw an error if your R session is not in interactive mode.

- "always" - `openFDA()` will always perform pagination. A list of **httr2** responses will be returned when pagination occurs.
- "never" - `openFDA()` will never perform pagination. Only the first **httr2** response will be returned.

`paging_verbosity`

A single string describing whether messages about pagination should be printed to the console. `openFDA()` uses the value of the package-level option `openFDA.paging_verbosity` for this argument, but you can set this behaviour on a per-function call level with this parameter. You can edit the option itself with `openFDA_options()`. Permissible values for `paging_verbosity` include:

- "verbose" - If paging can be performed, print a message to the console, stating how many requests are required with a minimum estimate for the time this will take.
- "quiet" - Even if paging can be performed, do not print messages about it to the console.

`handle_http_errors`

A single string defining how to handle HTTP codes other than 200 OK. `openFDA()` uses the value of the package-level option `openFDA.handle_http_errors` for this argument, but you can set this behaviour on a per-function call level with this parameter. You can edit the option itself with `openFDA_options()`. Permissible values for `handle_http_errors` include:

- "warn": If the returned HTTP code is not 200, issue a warning to the console and return the underlying **httr2** response object.
- "error": If the returned HTTP code is not 200, throw an (informative) error.
- "silent": If the returned HTTP code is not 200, return the underlying **httr2** response object without printing a warning.

Value

Either an **httr2** response object from `httr2::req_perform()` or list of these objects, depending on whether pagination was required. You can use `httr2::resp_body_json()` to extract JSON data from these responses.

Note

`openFDA()` caches its results with `memoise::memoise()`. Consequently, you should find that repeated, identical queries run faster than the first query, assuming you don't change your query parameters.

References

Kass-Hout TA, Xu Z, Mohebbi M, Nelsen H, Baker A, Levine J, Johansen E, Bright RA. **OpenFDA: an innovative platform providing access to a wealth of FDA's publicly available data** *J Am Med Inform Assoc* 2016, **23(3):596-600**. doi:10.1093/jamia/ocv153

See Also

Read the docs for [set_api_key\(\)](#) to learn how to set your openFDA API key. Check out [format_search_term\(\)](#) to see how input search vectors are converted to openFDA API searches.

Examples

```
if (httr2::secret_has_key("OPENFDA_KEY")) {
  set_api_key(httr2::secret_decrypt(
    "TEaDtqdFMq9_Montij5p9IY6T57IyqkbF8IYFV0pk-ttxotFUNdJSxgccAnkq4nQhplaf-r3deQ",
    "OPENFDA_KEY"
  ))

  resp <- openFDA(search = "openfda.manufacturer_name:gilead*",
    limit = 2,
    skip = 10,
    paging = "never",
    paging_verbosity = "quiet")

  # The function returns an `httr2` object
  print(resp)
}

# Bad inputs will cause informative errors - here, a bad API key is supplied
try(
  openFDA(search = "openfda.manufacturer_name:gilead*",
    api_key = "BAD_API_KEY",
    limit = 1)
)
```

openFDA_options

Get and set openFDA package-level options

Description

User-configurable options used by openFDA, which provide a mechanism for setting default behaviours.

Usage

```
openFDA_options(
  paging = NULL,
  paging_verbosity = NULL,
  handle_http_errors = NULL
)

current_openFDA_options()
```

Arguments

paging

A single string used to set the `openFDA.paging` option. The default, `NULL`, indicates that you don't want to change its value. The `openFDA.paging` option controls how `openFDA()` behaves when there are enough results that **paging is required**, and has the default value "ask". New values must be one of:

- "ask" - `openFDA()` will warn you that pagination is required and ask if you want this to be done. Depending on user input, either a single **httr2** response object or a list of **httr2** response objects will be returned. This will throw an error if your R session is not in interactive mode. When setting this option, the value must be one of:
- "always" - `openFDA()` will always perform pagination. A list of **httr2** responses will be returned when pagination occurs.
- "never" - `openFDA()` will never perform pagination. Only the first **httr2** response will be returned.

paging_verbosity

A single string used to set the `openFDA.paging_verbosity` option. The default, `NULL`, indicates that you don't want to change its value. The `openFDA.paging_verbosity` option controls whether `openFDA()` prints messages when paging is required, and has the default value "warn". New values must be one of:

- "verbose" - If paging can be performed, print a message to the console, stating how many requests are required with a minimum estimate for the time this will take.
- "quiet" - Print no messages to the console regarding paging.

handle_http_errors

A single string used to set the `openFDA.handle_http_errors` option. The default, `NULL`, indicates that you don't want to change its value. The `openFDA.handle_http_errors` option controls how `openFDA()` reacts when responses from the `openFDA` API have HTTP codes other than 200 OK, and has the default value "warn". New values must be one of:

- "warn": If the returned HTTP code is not 200, issue a warning to the console and return the underlying **httr2** response object.
- "error": If the returned HTTP code is not 200, throw an (informative) error.
- "silent": If the returned HTTP code is not 200, return the underlying **httr2** response object without printing a warning.

Value

For `openFDA_options()` returns a named character vector with current option values. This will include the updated option values, if any have been set.

For `current_openFDA_options()`, returns a named character vector with the current values for each `openFDA` package-level option.

Examples

```
# Print current options by leaving all input args as `NULL`:
```

```

openFDA_options()

# Or specify some options to set these:
openFDA_options(paging = "never", handle_http_errors = "silent")

# Make sure to give valid option values
try(openFDA_options(paging = "bad-value"))

# If you set an option using `options()` and it is not valid, you will
# invoke errors
try(
  rlang::with_options(openFDA.paging = "no", current_openFDA_options())
)

```

set_api_key

Get and set your openFDA API keys

Description

Get and set your openFDA API keys

Usage

```
set_api_key(api_key, user = "openFDA")
```

```
get_api_key(user = "openFDA")
```

Arguments

api_key	A single-length character vector with your openFDA API key. Do not pass the key in as a parameter in your scripts, though: see the 'Setting an openFDA API key' section below for more information. You can generate an API key on the FDA website .
user	A single-length character vector, used as the username argument in <code>keyring::key_set()</code> and <code>keyring::key_get()</code> , respectively. The default is "openFDA". Supply a custom value to this argument if for some reason you want to store multiple API keys (though note that the API restricts the number of requests on a per-IP address basis, as well as on a per-key basis).

Value

For `set_api_key()`, returns NULL invisibly. The function is called for its side effect: to store the openFDA API key securely using `keyring::key_set()`.

For `get_api_key()`, returns the API key as a single string. An error will be thrown if no key is found or if the returned key is an empty string ("").

Setting an openFDA API key

API keys are the sort of thing that should be stored securely. For that reason, `set_api_key()` and `get_api_key()` utilise the [keyring](#) package for storing/retrieving the API key. It is best practice not to set your API key by passing it directly to `set_api_key()`, or you risk exposing it either in your scripts or in `.Rhistory`.

Instead, start an interactive R session and run `set_api_key()`. R will prompt you to enter your API key, which will then be stored for future use without logging the key somewhere unsecure.

As for the key itself? It will be stored securely using `keyring::key_set()`. Once set on a machine, you shouldn't need to run `set_api_key()` again.

Creating a new keyring:

If your current [keyring backend](#) supports named keyrings, then when using `set_api_key()` for the first time you will be prompted to create a new keyring called "openFDA" using `keyring::keyring_create()`.

When asked to do this, you will need to enter a password which can be used to 'lock' and 'unlock' this new keyring (see `keyring::keyring_lock()` for details). However, you shouldn't need to use this password in the future, as the keyring will remain unlocked unless you use `keyring::keyring_lock()` to lock the "openFDA" keyring.

Examples

```
# Set your openFDA API key with `set_api_key()`
api_key <- "example_api_key"
set_api_key(api_key)

# Retrieve it with `get_api_key()`
get_api_key()

# An error will be thrown you try to set your API key to an empty string.
try(set_api_key(""))
```

Index

current_openFDA_options
 (openFDA_options), 8

format_search_term, 2
format_search_term(), 4, 8
format_sort_term, 3
format_sort_term(), 3

get_api_key (set_api_key), 10

httr2::req_perform(), 7
httr2::req_url(), 3, 4
httr2::resp_body_json(), 7

I(), 3, 4

keyring backend, 11

openFDA, 4
openFDA_options, 8

set_api_key, 10
set_api_key(), 8