Package 'HPZoneAPI'

April 9, 2026

Type Package

Title 'HPZone' API Interface

Version 1.3.0

Maintainer Aart Dijkstra <a.dijkstra@ggdnog.nl>

Description Package that simplifies the use of the 'HPZone' API. Most of the annoying and labor-intensive parts of the interface are handled by wrapper functions. Note that the API and its details are not publicly available. Information can be found at <https://www.ggdghorkennisnet.nl/groep/726-platform-infectieziekte-epidemiologen/documenten/map/9609> for those with access.

URL https://github.com/ggdatascience/HPZoneAPI

BugReports https://github.com/ggdatascience/HPZoneAPI/issues

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Imports dplyr, httr2, jsonlite, keyring, lubridate, magrittr, readxl, rstudioapi, safer, stringr

Depends R (>= 2.10)

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Aart Dijkstra [aut, cre, cph]

Repository CRAN

Date/Publication 2026-04-09 10:30:02 UTC

Contents

`HPZone_convert_dates` *Converts all relevant columns in the dataset to Date types.*

Description

Based on the available fields, a date for statistics is automatically generated, using the following logic: If cases: Date_of_onset > Datum_melding_aan_de_ggd > Case_creation_date. If situations: Start_date > Situation_creation_date If enquiries: Received_on > Date_closed

Usage

```
HPZone_convert_dates(
  data,
  search = "dat(e|um)|Received_on",
  statdate = "Date_stat"
)
```

Arguments

`data`	A dataset returned by the API.
`search`	Column names to search for. Default: anything containing 'date' or 'datum'.
`statdate`	Desired column name for the date for statistics field.

Value

An equivalent data.frame with Date types in the correct columns. Additionally, a column called 'Date_stat' is added, see Details.

Examples

```
## Not run:
HPZone_request("cases", "basic", where=c("Case_creation_date", ">", "2025-01-01")) |>
  HPZone_convert_dates()

## End(Not run)
```

`HPZone_fields` *Endpoints and fields available in the HPZone API*

Description

Endpoints and fields available in the HPZone API

Usage

```
HPZone_fields
```

Format

A tibble containing field names and metadata.

`HPZone_make_valid` *Takes a list of fields or endpoints and corrects them. This allows for short hand usage without having to check the documentation, e.g. 'Date of onset' instead of 'Date_of_onset'*

Description

Takes a list of fields or endpoints and corrects them. This allows for short hand usage without having to check the documentation, e.g. 'Date of onset' instead of 'Date_of_onset'

Usage

```
HPZone_make_valid(endpoints = NULL, fields = NULL)
```

Arguments

`endpoints`	A list of desired endpoints.
`fields`	A list of desired fields.

Value

A list of properly formatted fields or endpoints.

Examples

```
HPZone_make_valid("case") # should return "cases"
HPZone_make_valid(fields="case_creation") # should return "Case_creation_date"
HPZone_make_valid("case", fields=c("Family name", "Gp")) # should return c("Family_name", "Gp")
```

`HPZone_necessary_scope` *Determines the necessary scope for a given list of fields.*

Description

Determines the necessary scope for a given list of fields.

Usage

```
HPZone_necessary_scope(fields, endpoint = "Cases", resolve_fieldnames = TRUE)
```

Arguments

`fields`	List of desired fields from a query.
`endpoint`	The required endpoint.
`resolve_fieldnames`	Whether or not to parse the supplied fields and convert them to valid HPZone field names. (By calling [HPZone_make_valid])

Value

The required scope as expected by the HPZone API, i.e. "standard" or "extended".

Examples

```
# these variables do not required an extended scope; desired response = standard
HPZone_necessary_scope(c("Diagnosis", "Case_number", "Entered_by"))
# Family_name requires the extended scope; desired response = extended
HPZone_necessary_scope(c("Diagnosis", "Case_number", "Family_name"))
```

HPZone_request *Performs a HPZone request with the given parameters.*

Description

This function does most of the querybuilding for you, which allows for easier calling. There are several shorthands for field selection, the request is automatically paginated, and the necessary scope is automatically detected from the desired fields. Note that the take and skip elements are, by design, not present. If these are necessary, use [HPZone_request_query()] instead.

Usage

```
HPZone_request(endpoint, fields, where = NA, order = NA, verbose = FALSE)
```

Arguments

endpoint	The requested endpoint. Can be "cases", "situations", "enquiries", "contacts", or "actions". Case mismatch or spelling mismatch is automatically corrected with HPZone_make_valid().
fields	A vector containing the required fields. Spelling is automatically corrected. Alternatively, the keywords "all" (all available fields), "basic" (usual fields for surveillance), "standard" (only fields in the standard scope), or "none"/"id" (only HPZone ID and date) can be used. "basic" and "standard" can be combined: c("basic", "Longitude", "Latitude")
where	Either a vector containing pairs of 3 arguments, a literal query string, or a list outlining the selection criteria. See details.
order	A vector of field=order pairs, e.g. c("Case_creation_date"="ASC"). If no order is supplied, ASC is assumed.
verbose	Whether or not to display the calculated query and scope; useful for debugging query issues.

Details

The where clause can be specified in several formats. These can be: * A literal string containing GraphQL. E.g. "Status: { eq: \"Open\" }" * A vector of strings containing three pairs: field, comparator, value. E.g. c("Status", "=", "Open"). Any usual R-style comparators are allowed and automatically translated, GraphQL comparators are left as-is. * A list detailing the structure of the query, containing name-value pairs of keyword-selectors. This allows for complex and/or-structures, see the bottom example.

Value

An object containing the requested data points. This can be in different forms, depending on the request, but is simplified as much as possible.

See Also

[HPZone_request_paginated()], [HPZone_request_query()], [HPZone_make_valid()], [HPZone_convert_dates()]

Examples

```
## Not run:
# These statements are equal:
HPZone_request("cases", "all",
               where=c("Case_creation_date", ">", "2025-10-01"))
HPZone_request("cases", "all",
               where=c("Case_creation_date", "gt", "2025-10-01"))

# Selects cases after 2025-09-01, ordered by infection and then descending date.
HPZone_request("cases", "all",
               where=c("Case_creation_date", ">", "2025-09-01"),
               order=c("Infection", "Case_creation_date"="desc"))

# Selects all cases which were registered after 2025-01-01 AND where Infection equals Leptospirosis.
HPZone_request("cases", "all",
               where=list("and"=c("Case_creation_date", "gte", "2025-01-01",
                                  "Infection", "=", "Leptospirosis")))
# Note that the default is AND, so this statement is equal:
HPZone_request("cases", "all",
               where=c("Case_creation_date", "gte", "2025-01-01",
                       "Infection", "=", "Leptospirosis"))

# All cases after 2025-01-01 with either Leptospirosis or Malaria as infection.
# Note the nested list; adding a c() without a list() will warp the structure
# of the list and break everything.
HPZone_request("cases", "all",
               where=list(
                "and"=list(c("Case_creation_date", "gte", "2025-01-01"),
                           list("or"=c("Infection", "=", "Leptospirosis",
                                       "Infection", "==", "Malaria"))
                          )))

## End(Not run)
```

`HPZone_request_paginated`

Performs a HPZone request with the given parameters.

Description

Convenience wrapper around [HPZone_request_query()]. This function automatically pulls the available number of records, rather than only the first 500. Note that the current maximum for batching is 500 rows, so increasing n_max is not recommended. This function is mainly intended for use when the query builder in [HPZone_request()] is insufficient. Usage of [HPZone_request()] is considerably easier otherwise. An order clause is automatically added if not present to account for the lack of proper automatic sorting in the API.

Usage

```
HPZone_request_paginated(
  query,
  ...,
  n_max = 500,
  scope = API_env$scope_standard,
  verbose = FALSE
)
```

Arguments

query	A GraphQL query to send to the HPZone API. Note that keywords 'skip' and 'take' cannot be present in the query; this function will add them.
...	Parameters to be passed to sprintf(). If empty, the body is not passed through sprintf().
n_max	Maximum number of entries to request per call.
scope	The desired scope; either standard or extended.
verbose	Can be used to display information about the looping request.

Value

A data frame containing all the responses gathered from the API. Only the items will be returned.

See Also

[HPZone_request()], [HPZone_convert_dates()]

Examples

```
## Not run:
# Note the single quotes to facilitate double quote encapsulation for arguments.
HPZone_request_paginated(
  paste0('cases(where: {',
      'Case_creation_date: { gte: "2025-01-01" }',
    '}) {',
      'items { Case_creation_date, Case_number }',
    '}'))
# Or equal, making use of the sprintf integration:
startdate = "2025-01-01"
fields = c("Case_creation_date", "Case_number")
HPZone_request_paginated(
  'cases(where: { Case_creation_date: { gte: "%s" } }) { items { %s } }',
   startdate, stringr::str_c(fields, collapse=", ")
)

## End(Not run)
```

HPZone_request_query *Performs a HPZone request with the given parameters.*

Description

This function is a convenience wrapper around [HPZone_request_raw()], to facilitate easier coding. This function integrates [sprintf()] to allow for easier query design, quotes are automatically escaped, and the the result is unlisted to allow for easier access. See the example for differences with [HPZone_request_raw()]. Note that results are *not* automatically paginated; use [HPZone_request_paginated()] for that.

Usage

```
HPZone_request_query(query, ..., scope = API_env$scope_standard)
```

Arguments

query	A GraphQL query to send to the HPZone API. Note that only the actual query is required. (See the examples.)
...	Parameters to be passed to [sprintf()]. If empty, the body is not passed through [sprintf()].
scope	The desired scope; either standard or extended.

Value

An object containing the requested data points. This can be in different forms, depending on the request, but is simplified as much as possible.

See Also

[HPZone_request()], [HPZone_request_paginated()], [HPZone_convert_dates()]

Examples

```
# Note the difference between the raw and convenience functions.
# These lines are equal:
## Not run:
HPZone_request("cases", c("Case_creation_date", "Case_number"),
               where=c("Case_creation_date", ">=", "2025-01-01"))
HPZone_request_query(paste0('cases(where: {',
                            'Case_creation_date: { gte: "2025-01-01" }',
                           '}) {',
                            'items { Case_creation_date, Case_number }',
                           '}'))
HPZone_request_raw(paste0('{"query": "{ cases(where: {',
                            'Case_creation_date: { gte: \\"2025-01-01\\" }',
                          '}) {',
                            'items { Case_creation_date, Case_number }',
```

```
                          '} }"}'))

## End(Not run)
```

HPZone_request_raw *Performs a HPZone request with the given parameters.*

Description

Note that there are several helper functions that make the use of the API a lot simpler; specifically [HPZone_request()], [HPZone_request_paginated()]. This should only be used as a last resort; if manual GraphQL queries are required, [HPZone_request_query()] is highly advised instead.

Usage

```
HPZone_request_raw(body, scope = API_env$scope_standard)
```

Arguments

body	A GraphQL query to send to the HPZone API, including all necessary bracketing and JSON-elements.
scope	The desired scope; either standard or extended.

Value

An object containing the requested data points. This can be in different forms, depending on the request.

See Also

[HPZone_request()], [HPZone_request_paginated()], [HPZone_request_query()]

Examples

```
# Note the difference between the raw and convenience functions.
# These lines are equal:
## Not run:
HPZone_request("cases", c("Case_creation_date", "Case_number"),
               where=c("Case_creation_date", ">=", "2025-01-01"))
HPZone_request_query(paste0('cases(where: { ',
                      'Case_creation_date: { gte: "2025-01-01" }',
                      '})',
                      '{ items { Case_creation_date, Case_number } }')
                    )
HPZone_request_raw(paste0('{"query": "{ cases(where: {',
                            'Case_creation_date: { gte: \\"2025-01-01\\" }',
                          '})',
                          '{ items { Case_creation_date, Case_number } }',
                          '}"}'))
```

```
## End(Not run)
```

HPZone_setup *Initialisation function to define API credentials. This function must be called before anything else, as the details supplied in this call are required for use. Note that client_id and client_secret should preferably not be supplied in this call, but rather stored using [HPZone_store_credentials()].*

Description

Initialisation function to define API credentials. This function must be called before anything else, as the details supplied in this call are required for use. Note that client_id and client_secret should preferably not be supplied in this call, but rather stored using [HPZone_store_credentials()].

Usage

```
HPZone_setup(
  client_id = NA,
  client_secret = NA,
  standard = "standard",
  extended = "extended",
  token_url = "https://connect.govconext.nl/oidc/token",
  data_url = "https://api.hpzone.nl:8899/Edie"
)
```

Arguments

client_id	Client ID as supplied by InFact.
client_secret	Client secret as supplied by InFact.
standard	Name of the standard scope. Default: "standard"
extended	Name of the extended scope. Default: "extended"
token_url	Address of the token server.
data_url	Address of the data server.

Value

No return value.

Examples

```
# Not recommended:
HPZone_setup("id", "secret")

# Recommended:
## Not run:
HPZone_store_credentials() # call once
HPZone_setup() # will automatically read stored credentials

## End(Not run)
```

`HPZone_store_credentials`

Safely stores the HPZone API details, so you don't need to put them in your script. The default OS keyring backend is used through the keyring-package. Values are stored in an encrypted format to prevent harvesting by other applications. Note that the actual values are not supplied as arguments, but requested using rstudioapi password prompts.

Description

Safely stores the HPZone API details, so you don't need to put them in your script. The default OS keyring backend is used through the keyring-package. Values are stored in an encrypted format to prevent harvesting by other applications. Note that the actual values are not supplied as arguments, but requested using rstudioapi password prompts.

Usage

```
HPZone_store_credentials(client_id = TRUE, client_secret = TRUE)
```

Arguments

`client_id`	True / false: whether the client_id should be stored.
`client_secret`	True / false: whether the client_secret should be stored.

Value

N/A

Examples

```
## Not run:
# simply execute this line to store credentials
HPZone_store_credentials()
# after use, setup can be ran without arguments:
HPZone_setup()

## End(Not run)
```

`test_HPZone_token` *Tests the supplied HPZone token against the token endpoint. Note that this does not check the type of access, only the functionality of the token.*

Description

Tests the supplied HPZone token against the token endpoint. Note that this does not check the type of access, only the functionality of the token.

Usage

```
test_HPZone_token()
```

Value

No return value.

Examples

```
## Not run:
HPZone_setup()
# This will print the results.
test_HPZone_token()

## End(Not run)
```

Index