Package 'odns'

November 9, 2022

Title Access Scottish Health and Social Care Open Data

Version 1.0.2

Description Allows potential users of Scottish Health and Social Care Open Data (<https://www.opendata.nhs.scot/>) to easily explore and extract the available data.

License GPL (>= 3)

URL https://github.com/jrh-dev/odns

BugReports https://github.com/jrh-dev/odns/issues

Encoding UTF-8

RoxygenNote 7.2.1

Depends R (>= 3.5.0)

Imports glue, httr, utils, data.table, jsonlite

Suggests knitr, rmarkdown, testthat (>= 3.0.0), mockery, digest

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author James Hardy [aut, cre]

Maintainer James Hardy <jrh-dev@protonmail.com>

Repository CRAN

Date/Publication 2022-11-08 23:00:02 UTC

R topics documented:

`all_packages` *Details packages available from <opendata.nhs.scot>.*

Description

Details all packages available from <opendata.nhs.scot> in a data.frame along with the package id, with the option to limit results based on a search term.

Usage

```
all_packages(contains = NULL, limit = 1000L)
```

Arguments

`contains`	a character string containing an expression to be used as search criteria against the packages 'title' field.
`limit`	a numeric value specifying the maximum number of rows to be returned. Defaults to 1000L.

Value

a data.frame containing the names of all available packages and their package ids, or those whose name contains the string specified in the `contains` argument.

Examples

```
## Not run:
all_packages()
all_packages(contains = "standard-populations")

## End(Not run)
```

`all_resources` *Provides an overview of all resources available from <opendata.nhs.scot>.*

Description

Provides an overview of all resources available from <opendata.nhs.scot>, with the option to limit results based on both package and resource names. The returned data.frame can be used to look-up package and resource ids and is useful for exploring the available data sets.

Usage

```
all_resources(package_contains = NULL, resource_contains = NULL)
```

Arguments

`package_contains`
: a character string containing an expression to be used as search criteria against the packages 'title' field.

`resource_contains`
: a character string containing a regular expression to be matched against available resource names. If a character vector > length 1 is supplied, the first element is used.

Value

a data.frame containing details of all available packages and resources, or those containing the string specified in the `package_contains` and `resource_contains` arguments.

Examples

```
## Not run:
all_resources()
all_resources(package_contains = "standard-populations")
all_resources(
  package_contains = "standard-populations", resource_contains = "European"
)

## End(Not run)
```

cap_url *Produces error if input exceeds 2000 characters.*

Description

Used to ensure constructed URLs do not exceed 2000 characters.

Usage

```
cap_url(x)
```

Arguments

x	a character string to check.

Value

invisible.

detect_error *Detects http errors and provides enhanced details.*

Description

Detects http errors and provides enhanced details.

Usage

```
detect_error(result)
```

Arguments

result	A http response.

Value

invisible

get_data *Get data from a resource.*

Description

Get data from a resource in tabular format with the option to select fields and perform basic filtering. Where multiple data sets are required from a package and/or no field selection and filtering is required the `get_resource` function can be used.

Usage

```
get_data(resource, fields = NULL, limit = NULL, where = NULL, page_size = NULL)
```

Arguments

`resource`	A character string containing the resource id of the data set to be returned.
`fields`	A character vector containing the names of fields to be included in the returned table. The input is checked to ensure the specified fields exist in the chosen resource.
`limit`	An integer specifying the maximum number of records to be returned, the default NULL value returns all records.
`where`	A character string containing the 'WHERE' element of a simple SQL SELECT style query. Field names must be double quoted ", non-numeric values must be single quoted ', and both single and double quotes must be delimited. Example; `where = "\"AgeGroup\" = \'45-49 years\'"`.
`page_size`	An integer specifying the maximum number of records to be returned per query. Setting a value causes the use of offset pagination, multiple queries will be sent to return subsets of the available data. Subsets are joined before being returned. The default NULL value will always attempt to return all rows with a single query.

Value

A data.frame.

Examples

```
## Not run:
get_data(resource = "edee9731-daf7-4e0d-b525-e4c1469b8f69")

get_data(
  resource = "edee9731-daf7-4e0d-b525-e4c1469b8f69",
  fields = c("AgeGroup", "EuropeanStandardPopulation"),
  where = "\"AgeGroup\" = \'45-49 years\'"
)

## End(Not run)
```

get_resource *Get one or more resources or all resources within a package.*

Description

Get data from one or more resources, or all resources within a package, as a list, with each resource in tabular format. Where field selection and/or filtering of data is required the `get_data` function can be used.

Usage

```
get_resource(package = NULL, resource = NULL, limit = Inf)
```

Arguments

package	A character vector specifying package ids or names. If the `resource` argument is not provided all resources under each of the specified packages will be returned. The `package` argument itself is optional, but one of `package` or `resource` arguments must be provided.
resource	A character vector specifying resource ids or names. If the `package` argument is also provided then resources will only be returned if they exist under one of the specified packages, otherwise each of the specified resources will be returned. The `resource` argument itself is optional, but one of `resource` or `package` arguments must be provided.
limit	A numeric value specifying the maximum number of rows to be returned. Default value `Inf` returns all rows. Note; when multiple resources are returned the limit applies to each.

Value

A list containing all the resources within a package, or those specified, as data.frames.

Examples

```
## Not run:
get_resource(
  package = "4dd86111-7326-48c4-8763-8cc4aa190c3e",
  limit = 5L
  )

get_resource(
  package = "4dd86111-7326-48c4-8763-8cc4aa190c3e",
  resource = "edee9731-daf7-4e0d-b525-e4c1469b8f69",
  limit = 5L
  )

get_resource(
  package = "standard-populations",
```

```
  resource = "European Standard Population",
  limit = 5L
  )

 get_resource(
  resource = "European Standard Population",
  limit = 5L
  )

## End(Not run)
```

---

nrow_resource *Get the number of rows present in a resource.*

Description

Get the number of rows present in a resource.

Usage

```
nrow_resource(resource)
```

Arguments

resource	A character string containing the resource id of the data set to be returned.

Value

An integer of length 1 indicating the number of rows present in the specified resource.

Examples

```
## Not run:
nrow_resource(resource = "edee9731-daf7-4e0d-b525-e4c1469b8f69")

## End(Not run)
```

package_metadata *Get metadata for a package.*

Description

Get a specified packages metadata as a list.

Usage

```
package_metadata(package)
```

Arguments

package	A character vector of length 1 specifying a package id or name which identifies the package for which metadata should be returned.

Value

a list containing the package metadata.

Examples

```
## Not run:
package_metadata(package = "standard-populations")
package_metadata(package = "4dd86111-7326-48c4-8763-8cc4aa190c3e")

## End(Not run)
```

prep_nosql_query *Prepare an API query without SQL.*

Description

Prepare an API query without SQL.

Usage

```
prep_nosql_query(resource, fields, limit, offset)
```

Arguments

resource	A character string specifying resource id of the data set to be returned.
fields	A character vector specifying the names of fields to be included in the returned data.
limit	A numeric value specifying the maximum number of rows to be returned.
offset	A numeric value specifying the number of rows to skip.

Value

A character string containing the prepared query.

`prep_sql_query` *Prepare an API query with SQL.*

Description

Prepare an API query with SQL.

Usage

```
prep_sql_query(resource, fields, limit, offset, where)
```

Arguments

`resource`	a character string specifying resource id of the data set to be returned.
`fields`	a character vector specifying the names of fields to be included in the returned data.
`limit`	A numeric value specifying the maximum number of rows to be returned.
`offset`	A numeric value specifying the number of rows to skip.
`where`	A character string containing the 'WHERE' element of a simple SQL SELECT style query. Field names must be double quoted (`"}), non numeric values must be single quoted (\code{"`), and both single and double quotes must be delimited. Example; `where = "\"AgeGroup\" = \'45-49 years\'"`.

Value

A character string containing the prepared query.

`resource_data_items` *Get a table of available fields and their types for a specified resource.*

Description

Get a table of available fields and their types for a specified resource.

Usage

```
resource_data_items(resource)
```

Arguments

`resource`	A character string containing the resource id of the resource for which information is to be returned.

Value

A data.frame detailing the names and types of all fields available for the chosen resource.

Examples

```
## Not run:
resource_data_items(resource="edee9731-daf7-4e0d-b525-e4c1469b8f69")

## End(Not run)
```

`resource_metadata` *Get metadata for a resource.*

Description

Get a specified resources metadata as a list.

Usage

```
resource_metadata(resource)
```

Arguments

`resource`	A character vector of length 1 specifying a resource id which identifies the resource for which metadata should be returned.

Value

a list containing the resource metadata.

Examples

```
## Not run:
resource_metadata(resource = "edee9731-daf7-4e0d-b525-e4c1469b8f69")

## End(Not run)
```

valid_id *Basic check of whether the characters in a string is equal to 36*

Description

Basic check of whether the characters in a string is equal to 36

Usage

```
valid_id(x)
```

Arguments

x	A character string.

Value

logical value indicating whether the string checked consists of 36 characters.

Index