Package 'osdatahub'

July 22, 2025

Title Easier Interaction with the Ordnance Survey Data Hub

Version 0.3.0

Description Ordnance Survey ('OS') is the national mapping agency for Great Britain and produces a large variety of mapping and geospatial products. Much of OS's data is available via the OS Data Hub <https://osdatahub.os.uk/>, a platform that hosts both free and premium data products. 'osdatahub' provides a user-friendly way to access, query, and download these data.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.1

Suggests httptest, knitr, rmarkdown, sf, testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 2.10)

Imports geojsonsf, geos, httr, jsonlite

VignetteBuilder knitr

NeedsCompilation no

Author Chris Jochem [cre, aut],
Ordnance Survey Ltd [cph, fnd]

Maintainer Chris Jochem <chris.jochem@os.uk>

Repository CRAN

Date/Publication 2025-04-05 11:20:06 UTC

Contents

`bng_to_geom` *Return the geometry of a British National Grid square*

Description

Convert a valid British National Grid (BNG) grid reference string into a grid square with the resolution implied by the length of the reference string.

Usage

```
bng_to_geom(grid_ref, returnType = c("wkt", "geojson", "geos", "sf"))
```

Arguments

`grid_ref`	(character) BNG grid reference (required).
`returnType`	(character) Representation for the returned geometry. Choose `'wkt'` or `'geojson'` to return the geometry in Well-Known Text format or GeoJSON, respectively, `'geos'` to return an object of class `geos` or `'sf'` for a Simple Features object of class `sf`. Default is WKT format.

Details

The National Grid is a unique reference system that covers Great Britain in a series of grid squares at multiple scales. Grid references begin with 2 letters to identify 100km squares followed by a series of digits to identify quadrants nested within. For more information, see https://www.ordnancesurvey.co.uk/documents/resources/guide-to-nationalgrid.pdf

The purpose of this function is to generate geometries based on the extent of the grid square which can be used as spatial filters in OS Data Hub API queries.

Note that all geometries returned will have a coordinate reference system (CRS) of a EPSG:27700. The `sf` package must be installed in order to return an object of class `sf`.

Value

The coordinates of the grid square boundary in either Well-Known Text (WKT) format, GeoJSON format, an object of class `geos` or as a Simple Features object of class `sf`.

See Also

extent_from_bng()

Examples

```
bng_to_geom('TL63')
bng_to_geom('TL683365', returnType = 'geojson')
```

---

download_os_datapackages

Download OS premium data packages

Description

Main function for downloading OS data packages to your local machine.

Usage

```
download_os_datapackages(product, ...)

## S3 method for class 'package_list'
download_os_datapackages(
  product,
  file_name,
  output_dir,
  overwrite = FALSE,
  key = get_os_key(),
  ...
)

## S3 method for class 'data.frame'
download_os_datapackages(
  product,
  version,
  file_name,
  output_dir,
  overwrite = FALSE,
  key = get_os_key(),
  ...
)

## S3 method for class 'numeric'
download_os_datapackages(
  product,
  version,
  file_name,
```

```
  output_dir,
  overwrite = FALSE,
  key = get_os_key(),
  ...
)
```

Arguments

product	A product_list object retrieved and filtered using list_os_datapackages. Alternatively, a data.frame object or integer or character string of a product ID that can be filtered further.
...	Additional parameters. Not currently used.
file_name	(character) Filter downloads to only include those with this file name. Optional.
output_dir	Path to the directory where the downloaded files will be saved.
overwrite	Boolean. Should existing files be overwritten? Default is FALSE.
key	(character) OS API key. Default action is to search for an environment variable using get_os_key().
version	(numeric or character) Retrieve information on a specific version(s) of a data product. Required when product is a data.frame.

Details

The OS Downloads API assists with the discovery and download of OS OpenData and OS premium data packages. This function is used as the main step to download data packages to your local machine. It is designed to work best after list_os_datapackages is first used to search and filter for the specific download product. The package_list returned by the listing step can be used as the input value to download the desired files. Alternatively, it is possible to supply a product and version IDs directly when they are already known.

Before downloading a data package, it must be ordered online. See: https://osdatahub.os.uk/downloads/packages.

For more information on the Downloads API, see https://osdatahub.os.uk/docs/downloads/technicalSpecification.

Value

Silently returns the directory where the downloaded files are stored.

See Also

list_os_datapackages

Examples

```
## Not run:
# Search and filter available open products.
pkg_list <- list_os_datapackages()
```

```
# Use the package list to initiate a download.
# Note: 'version' will vary.
download_os_datapackages(pkg_list, version = 123, output_dir = tempdir())

## End(Not run)
```

`download_os_opendata` *Download OS OpenData Products*

Description

Main function for downloading OS open data product files to your local machine.

Usage

```
download_os_opendata(product, ...)

## S3 method for class 'product_list'
download_os_opendata(
  product,
  file_name,
  file_format,
  area,
  output_dir,
  overwrite = FALSE,
  ...
)

## S3 method for class 'character'
download_os_opendata(
  product,
  file_name,
  file_format,
  file_subformat,
  area,
  output_dir,
  overwrite = FALSE,
  ...
)
```

Arguments

`product`	A `product_list` object retrieved and filtered using `list_os_opendata`. Alternatively, a data product name as a character string.
`...`	Additional parameters. Not currently used.
`file_name`	(character) Filter downloads to only include those with this file name. Optional.

file_format	(character) Filter downloads to only include those with this format. Optional.
area	(character) Filter downloads for only this area. Use 'GB' for all Great Britain. Optional.
output_dir	Path to the directory where the downloaded files will be saved.
overwrite	Boolean. Should existing files be overwritten? Default is FALSE.
file_subformat	(character) Filter downloads to only include those with this subformat. Optional and only used when product is a character string.

Details

The OS Downloads API assists with the discovery and download of OS OpenData and OS Premium data packages. This function is used as the main step to download open data products to your local machine. It is designed to work best after list_os_opendata is first used to search and filter for the specific download product. The product_list returned by the listing step can be used as the input value to download the desired files. Alternatively, it is possible to supply a product name and filtering options based on file formats and areas.

The optional area filter is based on two-letter British National Grid tiles. Use 'GB' for all of Great Britain. Valid values area: GB, HP, HT, HU, HW, HX, HY, HZ, NA, NB, NC, ND, NF, NG, NH, NJ, NK, NL, NM, NN, NO, NR, NS, NT, NU, NW, NX, NY, NZ, OV, SD, SE, TA, SH, SJ, SK, TF, TG, SM, SN, SO, SP, TL, TM, SR, SS, ST, SU, TQ, TR, SV, SW, SX, SY, SZ, TV.

For more information on the Downloads API, see https://osdatahub.os.uk/docs/downloads/technicalSpecification.

Value

Silently returns the directory where the downloaded files are stored.

See Also

list_os_opendata()

Examples

```
# Search and filter available open products.
prod_list <- list_os_opendata('OpenGreenSpace',
                              file_format = 'GeoPackage',
                              area = 'GB')

# Use the product list to initiate a download.
download_os_opendata(prod_list, output_dir = tempdir())

# Combine search and download.
# Be sure to know the products to avoid downloading more data than desired.
download_os_opendata(product = 'OpenGreenSpace',
                     file_format = 'GeoPackage',
                     area = 'GB',
                     output_dir = tempdir())
```

extent *Create extents from geometries*

Description

Provide extents from various types of input features and geometries to be used as filters in OS Data Hub API queries.

Usage

```
extent_from_bbox(bbox, crs = "crs84", returnType = c("qExtent", "geos", "wkt"))

extent_from_polygon(
  polygon,
  crs = "crs84",
  returnType = c("qExtent", "geos", "wkt")
)

extent_from_geojson(
  geojson,
  crs = "crs84",
  returnType = c("qExtent", "geos", "wkt")
)

extent_from_radius(
  centre,
  radius,
  crs = "epsg:27700",
  returnType = c("qExtent", "geos", "wkt")
)

extent_from_bng(grid_ref, returnType = c("qExtent", "geos", "wkt"))
```

Arguments

bbox	A bounding box, passed as a numeric vector in (xmin,ymin,xmax,ymax) format, or a `data.frame` object with numeric columns.
crs	(character or numeric) The identifier for coordinate reference system information for the feature, either in the format "epsg:xxxx" or an EPSG number. e.g. British National Grid can be supplied as "epsg:27700" or 27700. Available CRS values are: EPSG:27700, EPSG:4326, EPSG:7405, EPSG:3857, and CRS84. Defaults to CRS84.
returnType	(character) Define the object returned. The default is `'qExtent'` to define a "query extent" object expected internally by `osdatahub`. Other options are `'wkt'` to return the geometry in Well-Known Text format or `'geos'` to return an object of class `geos`.

polygon	A polygon specified in a WKT string, an object of class geos, or an object of class sf.
geojson	A character string defining a polygon in GeoJSON format.
centre	Either a numeric vector with coordinates in the form (x, y), a Point object in a WKT string, a Point as a geos geometry or an object of class sf.
radius	(numeric) The radius of the circle in meters.
grid_ref	A character string with a British National Grid reference. The extent is formed by the grid square of the reference.

Details

When defining an extent by a radius around a point, the CRS must be either 'epsg:27700' or 'epsg:3857' which implies the units of the distance for the radius are meters.

Using crs='epsg:4326' implies that coordinates will be in Latitude/Longitude order. The equivalent projection with Longitude/Latitude order is 'crs84'.

The qExtent return option identifies a simple class of objects containing a polygon of the extent in WKT format, the bounding box coordinates, and a CRS string. It is intended to be used internally by functions in osdatahub.

Value

The coordinates of the polygon boundary as defined by returnType.

Examples

```
extent_from_bbox(c(600000, 310200, 600900, 310900), "epsg:27700", returnType = 'wkt')

# When using EPSG:4326, note the coordinate order expects latitude, longitude
extent_from_bbox(c(50.928731, -1.503346, 50.942376, -1.46762), crs="epsg:4326")

extent_from_radius(c(441317, 112165), radius = 200)

extent_from_bng("SU3715")
```

---

list_crs *Print the currently accepted EPSG codes.*

Description

Convenience function primarily used internally by osdatahub.

Usage

```
list_crs(...)
```

Arguments

...	Not currently used.

Value

(Invisible) Vector of character strings.

Examples

```
list_crs()
```

---

list_ngd_collections *Retrieve OS NGD Feature Collections*

Description

Query the osdatahub NGD Features API to gather information on available data collections. An API key is not required for this query.

Usage

```
list_ngd_collections(simple = TRUE)
```

Arguments

simple	(logical) Should only the collection ID be returned? Default is TRUE. Use FALSE to return the detailed output.

Details

OS NGD themes and collections have been created to group similar geographic entities and data types, making it quicker and easier to identify the data you need. The OGC API - Features standard also references feature collections, and in the context of OS NGD datasets, this is equivalent to feature types. The following naming convention has been applied to the feature collections: theme-collection-featuretype. Short codes have been used for both the theme and collection to keep the feature collection names manageable and not overly long. An example of the short codes used is: 'bld-fts-buildingline'. For more information, see https://osdatahub.os.uk/docs/ofa/technicalSpecification.

Value

If simple is TRUE then return a character vector of available collections identified by their shortened code, else return a data.frame with the full details.

Examples

```
ngd_collections <- list_ngd_collections(simple = TRUE)
ngd_collections[1:10]
```

`list_os_datapackages` *Retrieve information on premium OS data packages*

Description

Query the osdatahub Downloads API to gather information on available downloads for a specific OS premium data package based on given filters.

Usage

```
list_os_datapackages(product_id, version_id, key = get_os_key(), ...)
```

Arguments

`product_id`	(numeric or character) Retrieve information on a specific data product. Optional.
`version_id`	(numeric or character) Retrieve information on a specific version of a data product. Optional and only available when `product_id` has been specified.
`key`	(character) OS API key. Default action is to search for an environment variable using `get_os_key()`.
`...`	Additional paramters. Not currently used.

Details

The OS Downloads API assists with the discovery and download of OS OpenData and OS premium data packages. This function is used for initial listing and discovery of premium products. Use the product and version IDs from this list to filter further or to initiate a download.

Before downloading a data package, it must be ordered online. See: https://osdatahub.os.uk/downloads/packages.

When a `product_id` is not specified then all available data packages are listed. The `version_id` filter can be used to find the specific download, but this filter is only valid when a specific product has been specified first.

For more information on the Downloads API, see https://osdatahub.os.uk/docs/downloads/technicalSpecification.

Value

A `data.frame` or a `package_list`, which extends a `data.frame`, containing the information on downloadable files from the Downloads API.

See Also

download_os_datapackages()

Examples

```
## Not run:
# Retrieve a data.frame listing all OS Data Packages available.
# An API key is required and the packages must be ordered online first.
dp <- list_os_datapackages()

# Retrieve a specific data package.
# Note: 'product_id' will vary.
dp <- list_os_datapackages(product_id = 1234)

## End(Not run)
```

---

list_os_opendata *Retrieve information on OS OpenData Downloads*

Description

Query the osdatahub Downloads API to gather information on available data collections. An API key is not required to list OS OpenData.

Usage

```
list_os_opendata(product_id, file_name, file_format, file_subformat, area, ...)
```

Arguments

product_id	(character) Retrieve information on a specific data product. Optional.
file_name	(character) Filter downloads to only include those with this file name. Optional.
file_format	(character) Filter downloads to only include those with this format. Optional.
file_subformat	(character) Filter downloads to only include those with this subformat. Optional.
area	(character) Filter downloads for only this area. Use 'GB' for all Great Britain. Optional.
...	Additional paramters. Not currently used.

Details

The OS Downloads API assists with the discovery and download of OS OpenData and OS premium data packages. This function is used for initial listing and discovery of open data products. Use the product ID from this list to filter further or to initiate a download.

When a product_id is not specified then all available open data is listed. Additional filters (i.e. file_name, file_format, file_subformat, area) can be used to find the specific download, but these filters are only valid when a specific product has been specified first.

The optional area filter is based on two-letter British National Grid tiles. Use 'GB' for all of Great Britain. Valid values area: GB, HP, HT, HU, HW, HX, HY, HZ, NA, NB, NC, ND, NF, NG, NH, NJ, NK, NL, NM, NN, NO, NR, NS, NT, NU, NW, NX, NY, NZ, OV, SD, SE, TA, SH, SJ, SK, TF, TG, SM, SN, SO, SP, TL, TM, SR, SS, ST, SU, TQ, TR, SV, SW, SX, SY, SZ, TV.

For more information on the Downloads API, see https://osdatahub.os.uk/docs/downloads/technicalSpecification.

Value

A data.frame or a product_list, which extends a data.frame, containing the information on downloadable files from the Downloads API.

See Also

download_os_opendata()

Examples

```
# Retrieve a data.frame listing all OS OpenData products.
opendata <- list_os_opendata()
opendata[, c("name", "url")]
```

---

query_maps *Query the OS Maps API*

Description

Retrieve pre-rendered tiles from the web maps service of the Maps API in the Ordnance Survey Data Hub.

Usage

```
query_maps(
  x,
 layer = c("Road_27700", "Road_3857", "Outdoor_27700", "Outdoor_3857", "Light_27700",
    "Light_3857", "Leisure_27700"),
  zoom,
  output_dir,
  overwrite = FALSE,
  key = get_os_key(),
  ...
)

## S3 method for class 'qExtent'
query_maps(
  x,
```

```
  layer = c("Road_27700", "Road_3857", "Outdoor_27700", "Outdoor_3857", "Light_27700",
    "Light_3857", "Leisure_27700"),
  zoom,
  output_dir,
  overwrite = FALSE,
  key = get_os_key(),
  ...
)
```

Arguments

x	Object defining the query extent. Should be of type qExtent created by the extent_from_* functions.
layer	(character) The name of the layer to query. See details.
zoom	(numeric) The zoom level of the tiles to return. If omitted, a suitable zoom level will be estimated. See details.
output_dir	(character) Path to the directory where the downloaded tiles will be saved.
overwrite	Boolean. Should existing files be overwritten? Default is FALSE.
key	(character) OS API key. Default action is to search for an environment variable using get_os_key().
...	Additional parameters (not currently used).

Details

The OS Maps API serves pre-rendered raster tiles and is available in two projections; British National Grid and Web Mercator. This function provides basic access to download these tiles to your local machine.

Alternatively, you can request the maps using the Open Geospatial Consortium Web Map Tile Service (WMTS) standard or RESTful ZXY for easy access/visualisation in most GIS software and web mapping applications. More information on the Maps API is available from: https://osdatahub.os.uk/docs/wmts/technicalSpecification.

The parameter x currently only accepts a query extent created from extent_from_* family of functions. The coordinate reference system of this extent must match the coordinate reference system of the returned tiles (i.e. only EPSG:27700 and EPSG:3857 are accepted).

The available layers are Road, Outdoor, Light in both 27700 and 3857, plus Leisure in 27700. These should be specified as combined strings to the layer argument, e.g. 'Road_27700'.

The zoom levels available vary based on the projection of the tile matrix set. EPSG:3857 is from 7 to 20 and EPSG:27700 is from 0 to 13. See the technical specifications for more information on the scale and resolution: https://osdatahub.os.uk/docs/wmts/technicalSpecification

Value

A list of file paths to the downloaded image tiles, their bounding boxes, and coordinate reference system information.

See Also

extent

Examples

```
# Define an extent.
OS_ext <- extent_from_bng('SU3715')

# Download tiles.
imgTile <- query_maps(OS_ext,
                      layer = 'Light_27700',
                      output_dir = tempdir())

# The tiles can be merged together and georeferenced for spatial applications.
```

---

query_names *Query the OS Names API*

Description

Retrieve information from a geographic directory of identifiable places based on a free text search.

Usage

```
query_names(
  x,
  limit = 100,
  bounds,
  bbox_filter,
  local_type,
  key = get_os_key(),
  returnType = c("geojson", "list", "sf"),
  ...
)
```

Arguments

x	The free text search parameter.
limit	(numeric) The maximum number of features to return. Default is 100.
bounds	Biases the search results to a certain area. Should be of type qExtent created by the extent_from_* functions with CRS = EPSG:27700.
bbox_filter	Filters the results to a certain area. Should be of type qExtent created by the extent_from_* functions with CRS = EPSG:27700.
local_type	(character) Filters the results to certain local types. The available local types can be found at: https://osdatahub.os.uk/docs/names/technicalSpecification.

key	(character) OS API key. Default action is to search for an environment variable using get_os_key().
returnType	(character) Return the query results as the raw 'geojson', a nested 'list' object containing the returns, or convert them into Simple Features and return an object of class 'sf'.
...	Additional parameters (not currently used).

Details

The OS Names API is a geographic directory containing basic information about identifiable places. Those places are divided into themes, but the name of the place is the key property used in queries. The free text search is intended to be a "fuzzy" search.

Within OS Names, place names aren't unique. Extra location details are provided to help users refine their queries and accurately identify the named place they're interested in. These details include postcode district, populated place, district/borough, county/unitary authority, European region and country. Queries can also be refined by supplying bounding boxes or local types to search.

Technical details on the Names API are documented on the Data Hub: https://osdatahub.os.uk/docs/names/technicalSpecification.

Value

A GeoJSON string with the results of the API query, a list object, or an object of class sf based on the returnType parameter.

See Also

query_nearest_names(), extent

Examples

```
# Find names places by text search.
results <- query_names('Buckingham Palace', limit = 5)

# Use filters
results <- query_names('Southampton', local_type = 'City')

# Limit results to a bounding box
extent <- extent_from_bbox(c(600000, 310200, 600900, 310900),
                           crs = 'EPSG:27700')

results <- query_names('Norwich', bbox_filter = extent)
```

`query_nearest_names` *Query the OS Names API*

Description

Takes a pair of coordinates (X, Y) as an input to determine the closest name from a geographic directory of identifiable places.

Usage

```
query_nearest_names(
  point,
  radius = 100,
  local_type,
  key = get_os_key(),
  returnType = c("geojson", "list", "sf"),
  ...
)
```

Arguments

`point`	A set of British National Grid coordinates (EPSG:27700). Can be a set of coordinates as a numeric vector, an object of class `geos`, or an object of class `sf`.
`radius`	(numeric) The search radius in metres (max. 1000). Default is 100.
`local_type`	(character) Filters the results to certain local types. The available local types can be found at: https://osdatahub.os.uk/docs/names/technicalSpecification.
`key`	(character) OS API key. Default action is to search for an environment variable using `get_os_key()`.
`returnType`	(character) Return the query results as the raw `'geojson'`, a nested `'list'` object containing the returns, or convert them into Simple Features and return an object of class `'sf'`.
`...`	Additional parameters (not currently used).

Details

The OS Names API is a geographic directory containing basic information about identifiable places. Use this function to query Names to find the nearest named place to a given point location.

Within OS Names, place names aren't unique. Extra location details are provided to help users refine their queries and accurately identify the named place they're interested in. These details include postcode district, populated place, district/borough, county/unitary authority, European region and country. Queries can also be refined by supplying bounding boxes or local types to search.

Technical details on the Names API are documented on the Data Hub: https://osdatahub.os.uk/docs/names/technicalSpecification.

Value

A GeoJSON string with the results of the API query, a list object, or an object of class `sf` based on the `returnType` parameter.

See Also

`query_names()`, extent

Examples

```
# Named entity nearest to a point location
results <- query_nearest_names(point = c(440200,449300))
```

`query_nearest_places` *Query the OS Places API*

Description

Takes a pair of coordinates (X, Y)/(Lon, Lat) as an input to determine the closest address.

Usage

```
query_nearest_places(
  point,
  point_crs,
  radius = 100,
  output_crs = "EPSG:27700",
  classification_code,
  logical_status_code,
  dataset = "DPA",
  key = get_os_key(),
  returnType = c("geojson", "list", "sf"),
  ...
)
```

Arguments

`point`	A set of coordinates as a numeric vector, an object of class `geos`, or an object of class `sf`.
`point_crs`	(character or numeric) The identifier for coordinate reference system information for the point feature.
`radius`	(numeric) The search radius in metres (max. 1000). Default is 100.
`output_crs`	(character or numeric) The output CRS. Defaults to "EPSG:27700". Other options are EPSG:4326 or EPSG:3857.
`classification_code`	Classification codes to filter query by.

`logical_status_code`
: Logical status code to filter query by.

`dataset`
: (character) The dataset to return. Multiple values can be provided as a vector. Default is 'DPA'.

`key`
: (character) OS API key. Default action is to search for an environment variable using `get_os_key()`.

`returnType`
: (character) Return the query results as the raw `'geojson'`, a nested `'list'` object containing the returns, or convert them into Simple Features and return an object of class `'sf'`.

`...`
: Additional parameters (not currently used).

Details

The OS Places API provides a detailed view of an address and its life cycle. Use this function to query Places to find the address nearest to a given point location.

The Places API contains all the records of AddressBase® Premium and AddressBase® Premium – Islands and so provides all the information relating to an address or property from creation to retirement. It contains local authority, Ordnance Survey and Royal Mail® addresses, current addresses, and alternatives for current addresses, provisional addresses (such as planning developments) and historic information, plus OWPAs and cross references to the OS MasterMap® TOIDS®. OS Places API contains addresses located within the United Kingdom, Jersey, Guernsey and the Isle of Man. For address records in Jersey and Guernsey the coordinates will be '0.0' as they fall outside of the British National Grid. This means they are not compatible with the GeoSearch operations.

Technical details on the Places API are documented on the Data Hub: https://osdatahub.os.uk/docs/places/technicalSpecification.

Note: the Places API requires a Premium API key.

Value

A GeoJSON string with the results of the API query, a list object, or an object of class `sf` based on the `returnType` parameter.

See Also

`query_places()`, `query_postcode_places()`, `query_uprn_places()`

Examples

```
# Find address nearest to a point
pt <- c(437292.4, 115541.9)

results <- query_nearest_places(pt, point_crs = 'EPSG:27700')
```

query_ngd *Query the OS NGD Features API*

Description

Retrieve features from a given Collection of the National Geographic Database in the Ordnance Survey Data Hub.

Usage

```
query_ngd(x, ...)

## S3 method for class 'character'
query_ngd(
  x,
  collection,
  crs = "crs84",
  key = get_os_key(),
  returnType = c("geojson", "list", "sf"),
  ...
)

## S3 method for class 'qExtent'
query_ngd(
  x,
  collection,
  crs = "crs84",
  start_datetime,
  end_datetime,
  cql_filter,
  filter_crs,
  max_results = 100,
  offset = 0,
  key = get_os_key(),
  returnType = c("geojson", "list", "sf"),
  ...
)

## S3 method for class 'geos_geometry'
query_ngd(
  x,
  collection,
  crs = "crs84",
  start_datetime,
  end_datetime,
  cql_filter,
  filter_crs,
```

```
  max_results = 100,
  offset = 0,
  key = get_os_key(),
  returnType = c("geojson", "list", "sf"),
  ...
)

## S3 method for class 'sf'
query_ngd(
  x,
  collection,
  crs = "crs84",
  start_datetime,
  end_datetime,
  cql_filter,
  filter_crs,
  max_results = 100,
  offset = 0,
  key = get_os_key(),
  returnType = c("geojson", "list", "sf"),
  ...
)

## S3 method for class 'sfc'
query_ngd(
  x,
  collection,
  crs = "crs84",
  start_datetime,
  end_datetime,
  cql_filter,
  filter_crs,
  max_results = 100,
  offset = 0,
  key = get_os_key(),
  returnType = c("geojson", "list", "sf"),
  ...
)
```

Arguments

x	Object defining the query parameters, including feature IDs, extents, or spatial objects from which extents can be determined. If x is NULL or missing with other options specified by name, then the first max_results of the collection will be returned.
...	Additional parameters (not currently used).
collection	(character) The name of the NGD Collection to query (required). See list_ngd_collections().
crs	(character or numeric) The CRS for the returned features, either in the format

	"epsg:xxxx" or an EPSG number. e.g. British National Grid can be supplied as "epsg:27700" or 27700. Available CRS values are: EPSG:27700, EPSG:4326, EPSG:7405, EPSG:3857, and CRS84. Defaults to CRS84.
`key`	(character) OS API key. Default action is to search for an environment variable using `get_os_key()`.
`returnType`	(character) Return the query results as the raw `'geojson'`, a nested `'list'` object containing the returns, or convert them into Simple Features and return an object of class `'sf'`.
`start_datetime`	(datetime or string) Selects features that have a temporal property after the given start time. If you want to query a single timestamp, provide the same value to both `start_datetime` and `end_datetime`.
`end_datetime`	(datetime or string) Selects features that have a temporal property before the given end time. If you want to query a single timestamp, provide the same value to both `start_datetime` and `end_datetime`.
`cql_filter`	(character) A filter query in CQL format. More information about supported CQL operators can be found at https://osdatahub.os.uk/docs/ofa/technicalSpecification.
`filter_crs`	(character or numeric) The CRS for a given CQL query (if required), either in the format "epsg:xxxx" or an epsg number. e.g. British National Grid can be supplied as "epsg:27700" or 27700 Available CRS values are: EPSG:27700, EPSG:4326, EPSG:7405, EPSG:3857, and CRS84. Defaults to CRS84.
`max_results`	(numeric) The maximum number of features to return. Default is 100 which is the max return per page from the Data Hub.
`offset`	(numeric) The offset number skips past the specified number of features in the collection. Used to page through results. Default is 0.

Details

The value of `x` determines the type of query that is executed against the NGD API. When `x` is missing or set to `NULL` the first `n=max_results` features are returned. If a character string of an OSID is supplied as `x`, then that one feature from the collection will be returned.

When `x` is present `query_ngd()` will attempt to derive an extent from it. The `extent_from_*` family of functions are used and can be passed to `query_ngd` as a more verbose option. The one exception to this, `extent_from_grid_ref` must be used to create an extent and query a BNG grid reference.

The `start_datetime` and `end_datetime` parameters specify a valid date-time with UTC time zone (Z). Leave either empty to specify an open start/end interval. Only features that have a temporal geometry ('versionavailablefromdate' or 'versionavailabletodate') that intersect the value in the datetime parameter are selected. Example '2021-12-12T13:20:50Z'.

More information on the structure and data in the NGD is available from: https://osngd.gitbook.io/osngd/. Technical details on the NGD API are documented on the Data Hub: https://osdatahub.os.uk/docs/ofa/technicalSpecification.

Value

A GeoJSON string with the results of the API query, a list object, or an object of class `sf` based on the `returnType` parameter.

See Also

extent

Examples

```
# Return the first 50 features in the collection.
results <- query_ngd(collection = 'bld-fts-buildingline-1', max_results = 50)

# Return the most recent representation of a feature ID.
results <- query_ngd('0000013e-5fed-447d-a627-dae6fb215138',
                     collection = 'bld-fts-buildingline-1')

# Use a BNG reference to define a query extent.
results <- query_ngd(extent_from_bng("SU3715"),
                     collection = 'bld-fts-buildingpart-1')

# Add a temporal filter to query.
results <- query_ngd(collection = 'bld-fts-buildingline-1',
                     max_results = 50,
                     start_datetime = '2021-12-12 13:20:50')
```

---

query_places *Query the OS Places API*

Description

Retrieve information on UK addresses within a geographic area or based on a free text search.

Usage

```
query_places(x, ...)

## S3 method for class 'qExtent'
query_places(
  x,
  output_crs,
  limit = 100,
  classification_code,
  logical_status_code,
  dataset = "DPA",
  key = get_os_key(),
  returnType = c("geojson", "list", "sf"),
  ...
)

## S3 method for class 'character'
query_places(
```

```
  x,
  output_crs = "EPSG:27700",
  limit = 100,
  classification_code,
  logical_status_code,
  minmatch,
  matchprecision,
  dataset = "DPA",
  key = get_os_key(),
  returnType = c("geojson", "list", "sf"),
  ...
)
```

Arguments

x	Either a polygon created with `extent_from_*` functions defining the geographic area, or a character string to search.
...	Additional parameters (not currently used).
output_crs	Output CRS. Optional or will be defined by the extent.
limit	(numeric) The maximum number of features to return. Default is 100 which is the max return per page from the Data Hub.
classification_code	Classification codes to filter query by.
logical_status_code	Logical status code to filter query by.
dataset	(character) The dataset to return. Multiple values can be provided as a vector. Default is 'DPA'.
key	(character) OS API key. Default action is to search for an environment variable using `get_os_key()`.
returnType	(character) Return the query results as the raw `'geojson'`, a nested `'list'` object containing the returns, or convert them into Simple Features and return an object of class `'sf'`.
minmatch	The minimum matching score a result has to be returned.
matchprecision	The decimal point position at which the match score value is to be truncated.

Details

The OS Places API provides a detailed view of an address and its life cycle. Use this function to query Places based on a geographic area or a free text search.

The Places API contains all the records of AddressBase® Premium and AddressBase® Premium – Islands and so provides all the information relating to an address or property from creation to retirement. It contains local authority, Ordnance Survey and Royal Mail® addresses, current addresses, and alternatives for current addresses, provisional addresses (such as planning developments) and historic information, plus OWPAs and cross references to the OS MasterMap® TOIDS®. OS Places API contains addresses located within the United Kingdom, Jersey, Guernsey and the Isle of Man.

For address records in Jersey and Guernsey the coordinates will be '0.0' as they fall outside of the British National Grid. This means they are not compatible with the GeoSearch operations.

Technical details on the Places API are documented on the Data Hub: https://osdatahub.os.uk/docs/places/technicalSpecification.

Note: the Places API requires a Premium API key.

Value

A GeoJSON string with the results of the API query, a list object, or an object of class `sf` based on the `returnType` parameter.

See Also

extent, query_nearest_places(), query_postcode_places(), query_uprn_places()

Examples

```
# Addresses within a bounding box
extent <- extent_from_bbox(c(600000, 310200, 600900, 310900),
                           crs = 'EPSG:27700')

results <- query_places(extent, limit = 50)

# Find addresses by text search.
results <- query_places('Ordnance Survey, Adanac Drive, SO16',
                        minmatch = 0.5)
```

---

query_postcode_places *Query the OS Places API*

Description

A query of addresses based on a property's postcode.

Usage

```
query_postcode_places(
  postcode,
  output_crs = "EPSG:27700",
  limit = 100,
  classification_code,
  logical_status_code,
  dataset = "DPA",
  key = get_os_key(),
  returnType = c("geojson", "list", "sf"),
  ...
)
```

Arguments

`postcode`	The postcode search parameter as a character.
`output_crs`	(character or numeric) The output CRS. Defaults to "EPSG:27700". Other options are EPSG:4326 or EPSG:3857.
`limit`	(numeric) The maximum number of features to return. Default is 100 which is the max return per page from the Data Hub.
`classification_code`	Classification codes to filter query by.
`logical_status_code`	Logical status code to filter query by.
`dataset`	(character) The dataset to return. Multiple values can be provided as a vector. Default is 'DPA'.
`key`	(character) OS API key. Default action is to search for an environment variable using `get_os_key()`.
`returnType`	(character) Return the query results as the raw `'geojson'`, a nested `'list'` object containing the returns, or convert them into Simple Features and return an object of class `'sf'`.
`...`	Additional parameters (not currently used).

Details

The OS Places API provides a detailed view of an address and its life cycle. Use this function to query Places based on a postcode search. The minimum search parameter for this resource is the postcode area and postcode district. For example, 'SO16' is a valid search. Full postcodes, consisting of area, district, sector and unit, e.g. SO16 0AS can also be supplied.

The Places API contains all the records of AddressBase® Premium and AddressBase® Premium – Islands and so provides all the information relating to an address or property from creation to retirement. It contains local authority, Ordnance Survey and Royal Mail® addresses, current addresses, and alternatives for current addresses, provisional addresses (such as planning developments) and historic information, plus OWPAs and cross references to the OS MasterMap® TOIDS®. OS Places API contains addresses located within the United Kingdom, Jersey, Guernsey and the Isle of Man. For address records in Jersey and Guernsey the coordinates will be '0.0' as they fall outside of the British National Grid. This means they are not compatible with the GeoSearch operations.

Technical details on the Places API are documented on the Data Hub: https://osdatahub.os.uk/docs/places/technicalSpecification.

Note: the Places API requires a Premium API key.

Value

A GeoJSON string with the results of the API query, a list object, or an object of class `sf` based on the `returnType` parameter.

See Also

`query_places()`, `query_nearest_places()`, `query_uprn_places()`

Examples

```
results <- query_postcode_places(postcode = 'SO16 0AS')
```

`query_uprn_places` *Query the OS Places API*

Description

A query of addresses based on a property's UPRN.

Usage

```
query_uprn_places(
  uprn,
  output_crs = "EPSG:27700",
  classification_code,
  logical_status_code,
  dataset = "DPA",
  key = get_os_key(),
  returnType = c("geojson", "list", "sf"),
  ...
)
```

Arguments

`uprn`	A valid UPRN.
`output_crs`	(character or numeric) The output CRS. Defaults to "EPSG:27700". Other options are EPSG:4326 or EPSG:3857.
`classification_code`	Classification codes to filter query by.
`logical_status_code`	Logical status code to filter query by.
`dataset`	(character) The dataset to return. Multiple values can be provided as a vector. Default is 'DPA'.
`key`	(character) OS API key. Default action is to search for an environment variable using `get_os_key()`.
`returnType`	(character) Return the query results as the raw `'geojson'`, a nested `'list'` object containing the returns, or convert them into Simple Features and return an object of class `'sf'`.
`...`	Additional parameters (not currently used).

Details

The OS Places API provides a detailed view of an address and its life cycle. Use this function to query Places based on a UPRN search.

The Places API contains all the records of AddressBase® Premium and AddressBase® Premium – Islands and so provides all the information relating to an address or property from creation to retirement. It contains local authority, Ordnance Survey and Royal Mail® addresses, current addresses, and alternatives for current addresses, provisional addresses (such as planning developments) and historic information, plus OWPAs and cross references to the OS MasterMap® TOIDS®. OS Places API contains addresses located within the United Kingdom, Jersey, Guernsey and the Isle of Man. For address records in Jersey and Guernsey the coordinates will be '0.0' as they fall outside of the British National Grid. This means they are not compatible with the GeoSearch operations.

Technical details on the Places API are documented on the Data Hub: https://osdatahub.os.uk/docs/places/technicalSpecification.

Note: the Places API requires a Premium API key.

Value

A GeoJSON string with the results of the API query, a list object, or an object of class `sf` based on the `returnType` parameter.

See Also

`query_places()`, `query_nearest_places()`, `query_uprn_places()`

Examples

```
results <- query_uprn_places(uprn = 200010019924)
```

`set_os_key` *Set credentials for OS Data Hub*

Description

In order to use the Ordnance Survey Data Hub a valid API key is required.

Usage

```
set_os_key(apikey)

get_os_key()

has_os_key()
```

Arguments

`apikey`	(character) Required project API key.

Details

Stores the user provided character string in an environment variable named `OS_API_KEY`. No validation of the key is applied when storing. To obtain a key go to https://osdatahub.os.uk/.

Be careful not to reveal secrets including API keys. This function may print the API key to the console. It is used internally by the `osdatahub` query functions.

Primarily this is used internally to control when examples are executed.

Value

(Invisibly) A logical value from `Sys.setenv` whether an environment variable was set.

If an environment variable named `OS_API_KEY` is present, the character string for the variable is returned.

If an environment variable named `OS_API_KEY` is present, then `TRUE`, else this function returns `FALSE`.

Examples

```
set_os_key('my-api-key')


my_api_key <- get_os_key()

has_os_key()
```

Index