

Package ‘Catastro’

June 3, 2026

Title Interface to the Spanish 'Catastro' Web Services

Version 1.0.2

Description Access public spatial data from the Spanish 'Catastro' through its 'INSPIRE' and related web services. Retrieve parcel, building, address and map image data, and convert between parcel references and coordinates.

License GPL-2

URL <https://ropenspain.github.io/Catastro/>,
<https://github.com/rOpenSpain/Catastro>

BugReports <https://github.com/rOpenSpain/Catastro/issues>

Depends R (>= 4.1.0)

Imports cli, dplyr, httr2 (>= 1.0.0), lifecycle, mapSpain (>= 1.0.0), rappdirs (>= 0.3.0), sf (>= 1.0.0), terra, tibble, tools, utils, xml2

Suggests ggplot2, knitr, quarto, testthat (>= 3.0.0), tidyterra, withr

VignetteBuilder quarto

Config/Needs/website ropenspain/rostheme, devtools, sessioninfo, remotes, sfheaders, rapidjsonr, jsonify, geometries, magick

Config/roxygen2/markdown TRUE

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

Config/testthat/parallel true

Copyright © Dirección General del Catastro
<<https://www.catastro.meh.es/>>

Encoding UTF-8

LazyData true

X-schema.org-isPartOf <https://ropenspain.es/>

X-schema.org-keywords catastro, cran, cran-r, gis, maps, r, r-package, ropenspain, rstats, spain, spatial, static-tiles

NeedsCompilation no

Author Ángel Delgado Panadero [aut, cph] (ORCID: <https://orcid.org/0000-0002-8189-9251>),
 Iñaki Ucar [ctb] (ORCID: <https://orcid.org/0000-0001-6403-5550>),
 Diego Hernangómez [aut, cre] (ORCID: <https://orcid.org/0000-0001-8457-4658>)

Maintainer Diego Hernangómez <diego.hernangomezherrero@gmail.com>

Repository CRAN

Date/Publication 2026-06-03 19:10:02 UTC

Contents

catr_atom_get_address	2
catr_atom_get_address_db_all	4
catr_atom_get_buildings	6
catr_atom_get_buildings_db_all	7
catr_atom_get_parcel	9
catr_atom_get_parcel_db_all	11
catr_atom_search_munic	12
catr_clear_cache	13
catr_get_code_from_coords	14
catr_ovc_get_cod_munic	16
catr_ovc_get_cod_provinces	18
catr_ovc_get_cpmrc	19
catr_ovc_get_recoor	20
catr_ovc_get_recoor_distancia	21
catr_set_cache_dir	23
catr_srs_values	25
catr_wfs_get_address_bbox	26
catr_wfs_get_buildings_bbox	28
catr_wfs_get_parcel_bbox	30
catr_wms_get_layer	33
inspire_wfs_get	36

Index **38**

catr_atom_get_address *ATOM INSPIRE: download all addresses of a municipality*

Description

Retrieve the spatial data of all addresses belonging to a single municipality using the ATOM INSPIRE service. This function also returns corresponding street information in fields prefixed with `tfname_*`.

Usage

```

catr_atom_get_address(
  munic,
  to = NULL,
  cache = deprecated(),
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE
)

```

Arguments

munic	Municipality to extract, can be part of a string or a cadastral code. See catr_atom_search_munic() for getting the cadastral codes.
to	Optional argument for defining the territorial office to which munic belongs. This argument is a helper for narrowing the search.
cache	[Deprecated] cache is no longer supported, this function always caches results.
update_cache	Logical. Should the cached file be refreshed? Defaults to FALSE. When set to TRUE, it forces a new download.
cache_dir	Path to a cache directory. On NULL, the function stores cached files in a temporary directory (see base::tempdir()).
verbose	Logical. If TRUE, displays informational messages.

Value

A [sf](#) object.

References

[API documentation](#).

[INSPIRE services for cadastral cartography](#).

See Also

Related INSPIRE API functions: [catr_atom_get_address_db_all\(\)](#), [catr_atom_get_buildings\(\)](#), [catr_atom_get_buildings_db_all\(\)](#), [catr_atom_get_parcel\(\)](#), [catr_atom_get_parcel_db_all\(\)](#), [catr_wfs_get_address_bbox\(\)](#), [catr_wfs_get_buildings_bbox\(\)](#), [catr_wfs_get_parcel_bbox\(\)](#), [catr_wms_get_layer\(\)](#), [inspire_wfs_get\(\)](#)

Related ATOM INSPIRE functions: [catr_atom_get_address_db_all\(\)](#), [catr_atom_get_buildings\(\)](#), [catr_atom_get_buildings_db_all\(\)](#), [catr_atom_get_parcel\(\)](#), [catr_atom_get_parcel_db_all\(\)](#), [catr_atom_search_munic\(\)](#)

Other addresses: [catr_atom_get_address_db_all\(\)](#), [catr_wfs_get_address_bbox\(\)](#)

Other spatial: [catr_atom_get_buildings\(\)](#), [catr_atom_get_parcel\(\)](#), [catr_wfs_get_address_bbox\(\)](#), [catr_wfs_get_buildings_bbox\(\)](#), [catr_wfs_get_parcel_bbox\(\)](#), [catr_wms_get_layer\(\)](#)

Examples

```
s <- catr_atom_get_address("Melque", to = "Segovia")

library(ggplot2)

ggplot(s) +
  geom_sf(aes(color = specification)) +
  coord_sf(
    xlim = c(376200, 376850),
    ylim = c(4545000, 4546000)
  ) +
  labs(
    title = "Addresses",
    subtitle = "Melque de Cercos, Segovia"
  )
```

catr_atom_get_address_db_all

ATOM INSPIRE: reference database for ATOM addresses

Description

Create a database containing the URLs provided in the ATOM INSPIRE service of the Spanish Cadastre for extracting addresses.

`catr_atom_get_address_db_all()` provides a top-level table with all territorial offices, except the Basque Country and Navarre, and the municipalities included in each office. `catr_atom_get_address_db_to()` provides a table for one territorial office and its municipalities.

Usage

```
catr_atom_get_address_db_all(
  cache = deprecated(),
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE
)

catr_atom_get_address_db_to(
  to,
  cache = deprecated(),
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE
)
```

Arguments

cache	[Deprecated] cache is no longer supported, this function always caches results.
update_cache	Logical. Should the cached file be refreshed? Defaults to FALSE. When set to TRUE, it forces a new download.
cache_dir	Path to a cache directory. On NULL, the function stores cached files in a temporary directory (see <code>base::tempdir()</code>).
verbose	Logical. If TRUE, displays informational messages.
to	Character. Territorial office. Internally uses <code>base::grep()</code> for matching.

Value

A **tibble** with the information requested with the following fields:

- territorial_office: Territorial office, corresponding to each province of Spain except the Basque Country and Navarre.
- url: ATOM URL for the corresponding territorial office.
- munic: Name of the municipality.
- date: Reference date of the data. The information from this service is updated twice a year.

Source

<https://www.catastro.hacienda.gob.es/INSPIRE/Addresses/ES.SDGC.AD.atom.xml>

See Also

Related INSPIRE API functions: `catr_atom_get_address()`, `catr_atom_get_buildings()`, `catr_atom_get_buildings_db_all()`, `catr_atom_get_parcel()`, `catr_atom_get_parcel_db_all()`, `catr_wfs_get_address_bbox()`, `catr_wfs_get_buildings_bbox()`, `catr_wfs_get_parcel_bbox()`, `catr_wms_get_layer()`, `inspire_wfs_get()`

Related ATOM INSPIRE functions: `catr_atom_get_address()`, `catr_atom_get_buildings()`, `catr_atom_get_buildings_db_all()`, `catr_atom_get_parcel()`, `catr_atom_get_parcel_db_all()`, `catr_atom_search_munic()`

Other addresses: `catr_atom_get_address()`, `catr_wfs_get_address_bbox()`

Other databases: `catr_atom_get_buildings_db_all()`, `catr_atom_get_parcel_db_all()`, `catr_atom_search_munic()`, `catr_srs_values`

Examples

```
catr_atom_get_address_db_all()
```

`catr_atom_get_buildings`*ATOM INSPIRE: download all buildings of a municipality*

Description

Retrieve the spatial data of all buildings belonging to a single municipality using the ATOM INSPIRE service.

Usage

```
catr_atom_get_buildings(  
  munic,  
  to = NULL,  
  what = c("building", "buildingpart", "other"),  
  cache = deprecated(),  
  update_cache = FALSE,  
  cache_dir = NULL,  
  verbose = FALSE  
)
```

Arguments

<code>munic</code>	Municipality to extract, can be part of a string or a cadastral code. See catr_atom_search_munic() for getting the cadastral codes.
<code>to</code>	Optional argument for defining the territorial office to which <code>munic</code> belongs. This argument is a helper for narrowing the search.
<code>what</code>	Information to load. Options are: <ul style="list-style-type: none">• "building" for buildings.• "buildingpart" for parts of a building.• "other" for other elements such as swimming pools.
<code>cache</code>	[Deprecated] cache is no longer supported, this function always caches results.
<code>update_cache</code>	Logical. Should the cached file be refreshed? Defaults to FALSE. When set to TRUE, it forces a new download.
<code>cache_dir</code>	Path to a cache directory. On NULL, the function stores cached files in a temporary directory (see base::tempdir()).
<code>verbose</code>	Logical. If TRUE, displays informational messages.

Value

A [sf](#) object.

References

[API documentation](#).

[INSPIRE services for cadastral cartography](#).

See Also

Related INSPIRE API functions: `catr_atom_get_address()`, `catr_atom_get_address_db_all()`, `catr_atom_get_buildings_db_all()`, `catr_atom_get_parcelas()`, `catr_atom_get_parcelas_db_all()`, `catr_wfs_get_address_bbox()`, `catr_wfs_get_buildings_bbox()`, `catr_wfs_get_parcelas_bbox()`, `catr_wms_get_layer()`, `inspire_wfs_get()`

Related ATOM INSPIRE functions: `catr_atom_get_address()`, `catr_atom_get_address_db_all()`, `catr_atom_get_buildings_db_all()`, `catr_atom_get_parcelas()`, `catr_atom_get_parcelas_db_all()`, `catr_atom_search_munic()`

Other buildings: `catr_atom_get_buildings_db_all()`, `catr_wfs_get_buildings_bbox()`

Other spatial: `catr_atom_get_address()`, `catr_atom_get_parcelas()`, `catr_wfs_get_address_bbox()`, `catr_wfs_get_buildings_bbox()`, `catr_wfs_get_parcelas_bbox()`, `catr_wms_get_layer()`

Examples

```
s <- catr_atom_get_buildings("Nava de la Asuncion", to = "Segovia")

library(ggplot2)
ggplot(s) +
  geom_sf() +
  coord_sf(
    xlim = c(374500, 375500),
    ylim = c(4556500, 4557500)
  ) +
  labs(
    title = "Buildings",
    subtitle = "Nava de la Asuncion, Segovia"
  )
```

`catr_atom_get_buildings_db_all`

ATOM INSPIRE: reference database for ATOM buildings

Description

Create a database containing the URLs provided in the ATOM INSPIRE service of the Spanish Cadastre for extracting buildings.

`catr_atom_get_buildings_db_all()` provides a top-level table with all territorial offices, except the Basque Country and Navarre, and the municipalities included in each office. `catr_atom_get_buildings_db_to()` provides a table for one territorial office and its municipalities.

Usage

```

catr_atom_get_buildings_db_all(
  cache = deprecated(),
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE
)

catr_atom_get_buildings_db_to(
  to,
  cache = deprecated(),
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE
)

```

Arguments

cache	[Deprecated] cache is no longer supported, this function always caches results.
update_cache	Logical. Should the cached file be refreshed? Defaults to FALSE. When set to TRUE, it forces a new download.
cache_dir	Path to a cache directory. On NULL, the function stores cached files in a temporary directory (see <code>base::tempdir()</code>).
verbose	Logical. If TRUE, displays informational messages.
to	Character. Territorial office. Internally uses <code>base::grep()</code> for matching.

Value

A `tibble` with the information requested with the following fields:

- `territorial_office`: Territorial office, corresponding to each province of Spain except the Basque Country and Navarre.
- `url`: ATOM URL for the corresponding territorial office.
- `munic`: Name of the municipality.
- `date`: Reference date of the data. The information from this service is updated twice a year.

Source

<https://www.catastro.hacienda.gob.es/INSPIRE/buildings/ES.SDGC.BU.atom.xml>

See Also

Related INSPIRE API functions: `catr_atom_get_address()`, `catr_atom_get_address_db_all()`, `catr_atom_get_buildings()`, `catr_atom_get_parcelas()`, `catr_atom_get_parcelas_db_all()`, `catr_wfs_get_address_bbox()`, `catr_wfs_get_buildings_bbox()`, `catr_wfs_get_parcelas_bbox()`, `catr_wms_get_layer()`, `inspire_wfs_get()`

Related ATOM INSPIRE functions: [catr_atom_get_address\(\)](#), [catr_atom_get_address_db_all\(\)](#), [catr_atom_get_buildings\(\)](#), [catr_atom_get_parcel\(\)](#), [catr_atom_get_parcel_db_all\(\)](#), [catr_atom_search_munic\(\)](#)

Other buildings: [catr_atom_get_buildings\(\)](#), [catr_wfs_get_buildings_bbox\(\)](#)

Other databases: [catr_atom_get_address_db_all\(\)](#), [catr_atom_get_parcel_db_all\(\)](#), [catr_atom_search_munic\(\)](#), [catr_srs_values](#)

Examples

```
catr_atom_get_buildings_db_all()
```

```
catr_atom_get_parcel  ATOM INSPIRE: download all cadastral parcels of a municipality
```

Description

Retrieve the spatial data of all cadastral parcels belonging to a single municipality using the ATOM INSPIRE service.

Usage

```
catr_atom_get_parcel(
  munic,
  to = NULL,
  what = c("parcel", "zoning"),
  cache = deprecated(),
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE
)
```

Arguments

munic	Municipality to extract, can be part of a string or a cadastral code. See catr_atom_search_munic() for getting the cadastral codes.
to	Optional argument for defining the territorial office to which munic belongs. This argument is a helper for narrowing the search.
what	Information to load. Options are: <ul style="list-style-type: none"> • "parcel" for cadastral parcels. • "zoning" for cadastral zoning.
cache	[Deprecated] cache is no longer supported, this function always caches results.
update_cache	Logical. Should the cached file be refreshed? Defaults to FALSE. When set to TRUE, it forces a new download.

cache_dir	Path to a cache directory. On NULL, the function stores cached files in a temporary directory (see <code>base::tempdir()</code>).
verbose	Logical. If TRUE, displays informational messages.

Value

A `sf` object.

References

[API documentation](#).

[INSPIRE services for cadastral cartography](#).

See Also

Related INSPIRE API functions: `catr_atom_get_address()`, `catr_atom_get_address_db_all()`, `catr_atom_get_buildings()`, `catr_atom_get_buildings_db_all()`, `catr_atom_get_parcel_db_all()`, `catr_wfs_get_address_bbox()`, `catr_wfs_get_buildings_bbox()`, `catr_wfs_get_parcel_bbox()`, `catr_wms_get_layer()`, `inspire_wfs_get()`

Related ATOM INSPIRE functions: `catr_atom_get_address()`, `catr_atom_get_address_db_all()`, `catr_atom_get_buildings()`, `catr_atom_get_buildings_db_all()`, `catr_atom_get_parcel_db_all()`, `catr_atom_search_munic()`

Other parcels: `catr_atom_get_parcel_db_all()`, `catr_wfs_get_parcel_bbox()`

Other spatial: `catr_atom_get_address()`, `catr_atom_get_buildings()`, `catr_wfs_get_address_bbox()`, `catr_wfs_get_buildings_bbox()`, `catr_wfs_get_parcel_bbox()`, `catr_wms_get_layer()`

Examples

```
s <- catr_atom_get_parcel("Melque", to = "Segovia", what = "parcel")

library(ggplot2)

ggplot(s) +
  geom_sf() +
  labs(
    title = "Cadastral parcels",
    subtitle = "Melque de Cercos, Segovia"
  )
```

 catr_atom_get_parcel_s_db_all

ATOM INSPIRE: reference database for ATOM cadastral parcels

Description

Create a database containing the URLs provided in the ATOM INSPIRE service of the Spanish Cadastre for extracting cadastral parcels.

catr_atom_get_parcel_s_db_all() provides a top-level table with all territorial offices, except the Basque Country and Navarre, and the municipalities included in each office. catr_atom_get_parcel_s_db_to() provides a table for one territorial office and its municipalities.

Usage

```
catr_atom_get_parcel_s_db_all(
  cache = deprecated(),
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE
)
```

```
catr_atom_get_parcel_s_db_to(
  to,
  cache = deprecated(),
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE
)
```

Arguments

cache	[Deprecated] cache is no longer supported, this function always caches results.
update_cache	Logical. Should the cached file be refreshed? Defaults to FALSE. When set to TRUE, it forces a new download.
cache_dir	Path to a cache directory. On NULL, the function stores cached files in a temporary directory (see <code>base::tempdir()</code>).
verbose	Logical. If TRUE, displays informational messages.
to	Character. Territorial office. Internally uses <code>base::grep()</code> for matching.

Value

A [tibble](#) with the information requested with the following fields:

- territorial_office: Territorial office, corresponding to each province of Spain except the Basque Country and Navarre.
- url: ATOM URL for the corresponding territorial office.

- `munic`: Name of the municipality.
- `date`: Reference date of the data. The information from this service is updated twice a year.

Source

<https://www.catastro.hacienda.gob.es/INSPIRE/CadastralParcels/ES.SDGC.CP.atom.xml>

See Also

Related INSPIRE API functions: `catr_atom_get_address()`, `catr_atom_get_address_db_all()`, `catr_atom_get_buildings()`, `catr_atom_get_buildings_db_all()`, `catr_atom_get_parcel()`, `catr_wfs_get_address_bbox()`, `catr_wfs_get_buildings_bbox()`, `catr_wfs_get_parcel_bbox()`, `catr_wms_get_layer()`, `inspire_wfs_get()`

Related ATOM INSPIRE functions: `catr_atom_get_address()`, `catr_atom_get_address_db_all()`, `catr_atom_get_buildings()`, `catr_atom_get_buildings_db_all()`, `catr_atom_get_parcel()`, `catr_atom_search_munic()`

Other parcels: `catr_atom_get_parcel()`, `catr_wfs_get_parcel_bbox()`

Other databases: `catr_atom_get_address_db_all()`, `catr_atom_get_buildings_db_all()`, `catr_atom_search_munic()`, `catr_srs_values`

Examples

```
catr_atom_get_parcel_db_all()
```

```
catr_atom_search_munic
```

ATOM INSPIRE: search for municipality codes

Description

Search for a municipality (as a string, part of a string, or code) and get the corresponding code according to the Cadastre.

Usage

```
catr_atom_search_munic(  
    munic,  
    to = NULL,  
    cache = deprecated(),  
    update_cache = FALSE,  
    cache_dir = NULL,  
    verbose = FALSE  
)
```

Arguments

munic	Municipality to extract, can be part of a string or a cadastral code. See catr_atom_search_munic() for getting the cadastral codes.
to	Optional argument for defining the territorial office to which munic belongs. This argument is a helper for narrowing the search.
cache	[Deprecated] cache is no longer supported, this function always caches results.
update_cache	Logical. Should the cached file be refreshed? Defaults to FALSE. When set to TRUE, it forces a new download.
cache_dir	Path to a cache directory. On NULL, the function stores cached files in a temporary directory (see base::tempdir()).
verbose	Logical. If TRUE, displays informational messages.

Value

A [tibble](#).

See Also

Related ATOM INSPIRE functions: [catr_atom_get_address\(\)](#), [catr_atom_get_address_db_all\(\)](#), [catr_atom_get_buildings\(\)](#), [catr_atom_get_buildings_db_all\(\)](#), [catr_atom_get_parcel\(\)](#), [catr_atom_get_parcel_db_all\(\)](#)

Other search: [catr_get_code_from_coords\(\)](#), [catr_ovc_get_cod_munic\(\)](#), [catr_ovc_get_cod_provinces\(\)](#)

Other databases: [catr_atom_get_address_db_all\(\)](#), [catr_atom_get_buildings_db_all\(\)](#), [catr_atom_get_parcel_db_all\(\)](#), [catr_srs_values](#)

Examples

```
catr_atom_search_munic("Mad")
```

catr_clear_cache	<i>Clear your Rhrefhttps://CRAN.R-project.org/package=CatastRo CatastRo cache directory</i>
------------------	--

Description

Use this function with caution. This function clears your cached data and configuration, specifically:

- Deletes the **CatastRo** config directory (`tools::R_user_dir("CatastRo", "config")`).
- Deletes the `cache_dir` directory.
- Deletes the values stored on `Sys.getenv("CATASTROESP_CACHE_DIR")`.

Usage

```
catr_clear_cache(config = FALSE, cached_data = TRUE, verbose = FALSE)
```

Arguments

config	If TRUE, deletes the configuration folder of CatastRo .
cached_data	If TRUE, deletes your cache_dir and all its contents.
verbose	Logical. If TRUE, displays informational messages.

Details

This function resets your cache state as if you had never installed and/or used **CatastRo**.

Value

Invisible. This function is called for its side effects.

See Also

[tools::R_user_dir\(\)](#)

Other cache utilities: [catr_set_cache_dir\(\)](#)

Examples

```
# Don't run this! It modifies your current state
## Not run:
my_cache <- catr_detect_cache_dir()

# Set an example cache
ex <- file.path(tempdir(), "example", "cache")
catr_set_cache_dir(ex, verbose = FALSE)

# Restore initial cache
catr_clear_cache(verbose = TRUE)

catr_set_cache_dir(my_cache)
identical(my_cache, catr_detect_cache_dir())

## End(Not run)
```

catr_get_code_from_coords

Get the cadastral municipality code from coordinates

Description

Get the municipality code for coordinates using a [sf](#) object or a pair of coordinates via [catr_ovc_get_cod_munic\(\)](#).

Usage

```
catr_get_code_from_coords(
  x,
  srs = NULL,
  verbose = FALSE,
  cache_dir = NULL,
  ...
)
```

Arguments

x	Can be one of: <ul style="list-style-type: none"> • A pair of coordinates $c(x, y)$. In this case the <code>srs</code> of the coordinates must be provided. • A <code>sf</code> object. If the object has several geometries, only the first geometry is used. This function extracts coordinates using <code>sf::st_centroid(x, of_largest_polygon = TRUE)</code>.
srs	SRS/CRS to use in the query. To see allowed values, use <code>catr_srs_values</code> , specifically the <code>ovc_service</code> column.
verbose	Logical. If TRUE, displays informational messages.
cache_dir	Path to a cache directory. On NULL, the function stores cached files in a temporary directory (see <code>base::tempdir()</code>).
...	Arguments passed on to <code>mapSpain::esp_get_munic_siane</code>
year	character string or number. Release year, it must be in formats YYYY (assuming end of year) or YYYY-MM-DD. Historical information starts as of 2005.
resolution	character string or number. Resolution of the geospatial data. One of: <ul style="list-style-type: none"> • "10": 1:10 million. • "6.5": 1:6.5 million. • "3": 1:3 million.
region	Optional. A vector of region names, NUTS or ISO codes (see <code>esp_dict_region_code()</code>).
munic	character string. A name or <code>regex</code> expression with the names of the required municipalities. NULL will return all municipalities.

Details

On a successful query, this function returns a `tibble` with one row including the following columns:

- `munic`: Name of the municipality according to the Cadastre.
- `catr_to`: Cadastral territorial office code.
- `catr_munic`: Municipality code as recorded on the Cadastre.
- `catrcode`: Full Cadastral code for the municipality.
- `cpro`: Province code according to the INE.

- `cmun`: Municipality code according to the INE.
- `inecode`: Full INE code for the municipality.
- Remaining fields: Check the API documentation.

Value

A [tibble](#). See **Details**.

See Also

`mapSpain::esp_get_munic_siane()`, `catr_ovc_get_cod_munic()`, `sf::st_centroid()`.

Other search: `catr_atom_search_munic()`, `catr_ovc_get_cod_munic()`, `catr_ovc_get_cod_provinces()`

Examples

```
# Use with coordinates
catr_get_code_from_coords(c(-16.25462, 28.46824), srs = 4326)
```

```
# Use with sf
prov <- mapSpain::esp_get_prov("Caceres")
catr_get_code_from_coords(prov)
```

`catr_ovc_get_cod_munic`

OVCCallejero: extract the code of a municipality

Description

Implementation of the OVCCallejero service [ConsultaMunicipioCodigos](#). Returns names and codes of a municipality according to the Cadastre and the INE (National Statistics Institute).

Usage

```
catr_ovc_get_cod_munic(cpro, cmun = NULL, cmun_ine = NULL, verbose = FALSE)
```

Arguments

<code>cpro</code>	The code of a province, as provided by <code>catr_ovc_get_cod_provinces()</code> .
<code>cmun</code> , <code>cmun_ine</code>	Code of a municipality, as recorded on the Spanish Cadastre (<code>cmun</code>) or the National Statistics Institute. Either <code>cmun</code> or <code>cmun_ine</code> must be provided.
<code>verbose</code>	Logical. If TRUE, displays informational messages.

Details

On a successful query, this function returns a [tibble](#) with one row including the following columns:

- `munic`: Name of the municipality according to the Cadastre.
- `catr_to`: Cadastral territorial office code.
- `catr_munic`: Municipality code as recorded on the Cadastre.
- `catrcode`: Full Cadastral code for the municipality.
- `cpro`: Province code according to the INE.
- `cmun`: Municipality code according to the INE.
- `inecode`: Full INE code for the municipality.
- Remaining fields: Check the API documentation.

Value

A [tibble](#). See **Details**.

References

[ConsultaMunicipioCodigos](#).

See Also

[mapSpain::esp_get_munic_siane\(\)](#) to get shapes of municipalities, including the INE code.

Related OVCCallejero functions: [catr_ovc_get_cod_provinces\(\)](#)

Other search: [catr_atom_search_munic\(\)](#), [catr_get_code_from_coords\(\)](#), [catr_ovc_get_cod_provinces\(\)](#)

Examples

```
# Get municipality by cadastral code
ab <- catr_ovc_get_cod_munic(cpro = 2, cmun = 900)

ab

# Same query using the INE code
ab2 <- catr_ovc_get_cod_munic(cpro = 2, cmun_ine = 3)

ab2
```

`catr_ovc_get_cod_provinces`*OVCCallejero: extract provinces with their codes*

Description

Implementation of the OVCCallejero service [ConsultaProvincia](#). Returns a list of provinces included in the Spanish Cadastre.

Usage

```
catr_ovc_get_cod_provinces(verbose = FALSE)
```

Arguments

`verbose` Logical. If TRUE, displays informational messages.

Value

A [tibble](#).

References

[ConsultaProvincia](#).

See Also

Related OVCCallejero functions: [catr_ovc_get_cod_munic\(\)](#)

Other search: [catr_atom_search_munic\(\)](#), [catr_get_code_from_coords\(\)](#), [catr_ovc_get_cod_munic\(\)](#)

Examples

```
catr_ovc_get_cod_provinces()
```

catr_ovc_get_cpmrc *OVCCoordenadas: geocode a cadastral reference*

Description

Implementation of the OVCCoordenadas service [Consulta CPMRC](#). Returns coordinates for a specific cadastral reference.

Usage

```
catr_ovc_get_cpmrc(  
  rc,  
  srs = 4326,  
  province = NULL,  
  municipality = NULL,  
  verbose = FALSE  
)
```

Arguments

rc	The cadastral reference to be geocoded.
srs	SRS/CRS to use in the query. To see allowed values, use catr_srs_values , specifically the ovc_service column.
province, municipality	Optional, used for narrowing the search.
verbose	Logical. If TRUE, displays informational messages.

Details

When the API does not provide any result, this function returns a [tibble](#) with the input arguments only.

On a successful query, this function returns a [tibble](#) with one row per cadastral reference, including the following columns:

- xcoord, ycoord: X and Y coordinates in the specified SRS.
- refcat: Cadastral reference.
- address: Address as recorded in the Cadastre.
- Remaining fields: Check the API documentation.

Value

A [tibble](#). See **Details**.

References

[Consulta CPMRC](#).

See Also

[catr_srs_values](#), vignette("ovcservice", package = "Catastro")

Related OVCCoordenadas functions: [catr_ovc_get_rccoor\(\)](#), [catr_ovc_get_rccoor_distancia\(\)](#), [catr_srs_values](#)

Other cadastral references: [catr_ovc_get_rccoor\(\)](#), [catr_ovc_get_rccoor_distancia\(\)](#)

Examples

```
# Using all arguments
catr_ovc_get_cpmrc("13077A01800039",
  4230,
  province = "CIUDAD REAL",
  municipality = "SANTA CRUZ DE MUDELA"
)

# Only the cadastral reference
catr_ovc_get_cpmrc("9872023VH5797S")
```

catr_ovc_get_rccoor *OVCCoordenadas: reverse geocode a cadastral reference*

Description

Implementation of the OVCCoordenadas service **Consulta RCCOOR**. Returns the cadastral reference found for a set of specific coordinates.

Usage

```
catr_ovc_get_rccoor(lat, lon, srs = 4326, verbose = FALSE)
```

Arguments

lat	Latitude for the query, expressed in the CRS/SRS defined by srs.
lon	Longitude for the query, expressed in the CRS/SRS defined by srs.
srs	SRS/CRS to use in the query. To see allowed values, use catr_srs_values , specifically the ovc_service column.
verbose	Logical. If TRUE, displays informational messages.

Details

When the API does not provide any result, the function returns a [tibble](#) with the input arguments only.

On a successful query, this function returns a [tibble](#) with one row per cadastral reference, including the following columns:

- `geo.xcen`, `geo.ycen`, `geo.srs`: Input arguments of the query.
- `refcat`: Cadastral reference.
- `address`: Address as recorded in the Cadastre.
- Remaining fields: Check the API documentation.

Value

A [tibble](#). See **Details**.

References

[Consulta RCCOOR](#).

See Also

[catr_srs_values](#), `vignette("ovcservice", package = "CatastRo")`

Related OVCCoordenadas functions: [catr_ovc_get_cpmrc\(\)](#), [catr_ovc_get_rccoor_distancia\(\)](#), [catr_srs_values](#)

Other cadastral references: [catr_ovc_get_cpmrc\(\)](#), [catr_ovc_get_rccoor_distancia\(\)](#)

Examples

```
catr_ovc_get_rccoor(  
  lat = 38.6196566583596,  
  lon = -3.45624183836806,  
  srs = 4326  
)
```

`catr_ovc_get_rccoor_distancia`

OVCCoordenadas: reverse geocode cadastral references near coordinates

Description

Implementation of the OVCCoordenadas service [Consulta RCCOOR Distancia](#). Returns cadastral references for coordinates. If none found, the API returns references in a 50 square meter area around the requested coordinates.

Usage

```
catr_ovc_get_rccoor_distancia(lat, lon, srs = 4326, verbose = FALSE)
```

Arguments

lat	Latitude for the query, expressed in the CRS/SRS defined by srs.
lon	Longitude for the query, expressed in the CRS/SRS defined by srs.
srs	SRS/CRS to use in the query. To see allowed values, use catr_srs_values , specifically the ovc_service column.
verbose	Logical. If TRUE, displays informational messages.

Details

When the API does not provide any result, the function returns a [tibble](#) with the input arguments only.

On a successful query, this function returns a [tibble](#) with one row per cadastral reference, including the following columns:

- geo.xcen, geo.ycen, geo.srs: Input arguments of the query.
- refcat: Cadastral reference.
- address: Address as recorded in the Cadastre.
- cmun_ine: Municipality code as registered on the INE (National Statistics Institute).
- Remaining fields: Check the API documentation.

Value

A [tibble](#). See **Details**.

References

[Consulta RCCOOR Distancia](#).

See Also

[catr_srs_values](#), `vignette("ovcservice", package = "Catastro")`

Related OVCCoordenadas functions: [catr_ovc_get_cpmrc\(\)](#), [catr_ovc_get_rccoor\(\)](#), [catr_srs_values](#)

Other cadastral references: [catr_ovc_get_cpmrc\(\)](#), [catr_ovc_get_rccoor\(\)](#)

Examples

```
catr_ovc_get_rccoor_distancia(  
  lat = 40.963200,  
  lon = -5.671420,  
  srs = 4326  
)
```

catr_set_cache_dir	<i>Set your R https://CRAN.R-project.org/package=CatastRo cache directory</i>
--------------------	---

Description

Store your `cache_dir` path locally for future sessions. Type `Sys.getenv("CATASTROESP_CACHE_DIR")` or use `catr_detect_cache_dir()` to find your cached path.

Usage

```
catr_set_cache_dir(  
  cache_dir = NULL,  
  overwrite = FALSE,  
  install = FALSE,  
  verbose = TRUE  
)  
  
catr_detect_cache_dir()
```

Arguments

<code>cache_dir</code>	Path to a cache directory. On <code>NULL</code> , the function stores cached files in a temporary directory (see <code>base::tempdir()</code>).
<code>overwrite</code>	Logical. If <code>TRUE</code> , overwrites an existing <code>CATASTROESP_CACHE_DIR</code> value already present on your machine.
<code>install</code>	Logical. If <code>TRUE</code> , installs the key on your local machine for use in future sessions. Defaults to <code>FALSE</code> . If <code>cache_dir</code> is <code>FALSE</code> , this argument is set to <code>FALSE</code> automatically.
<code>verbose</code>	Logical. If <code>TRUE</code> , displays informational messages.

Details

By default, when no `cache_dir` is set, the package uses a folder inside `base::tempdir()` (so files are temporary and are removed when the R session ends). To persist a cache across R sessions, use `catr_set_cache_dir(cache_dir, install = TRUE)` which writes the chosen path to a small configuration file under `tools::R_user_dir("CatastRo", "config")`.

Value

`catr_set_cache_dir()` returns an (invisible) character with the path to your `cache_dir`, but it is mainly called for its side effect.

`catr_detect_cache_dir()` returns the path to the `cache_dir` used in this session.

Caching strategies

Some files can be read from their online source without caching using the option `cache = FALSE`. Otherwise the source file is downloaded to your computer. **CatastRo** implements the following caching options:

- For occasional use, rely on the default `tempdir()`-based cache (no install).
- Modify the cache for a single session by setting `catr_set_cache_dir(cache_dir = "a/path/here")`.
- For reproducible workflows, install a persistent cache with `catr_set_cache_dir(cache_dir = "a/path/here", install = TRUE)`. This cache is kept across R sessions.
- For caching specific files, use the `cache_dir` argument in the corresponding function.

Sometimes cached files may be corrupt. In that case, try re-downloading the data by setting `update_cache = TRUE` in the corresponding function.

If you experience any problem downloading, try downloading the corresponding file by another method and save it in your `cache_dir`. Use the option `verbose = TRUE` to debug the API query and `catr_detect_cache_dir()` to identify your cached path.

Note

In **CatastRo** \geq 1.0.0 the location of the configuration file has moved from `rappdirs::user_config_dir("CatastRo", "R")` to `tools::R_user_dir("CatastRo", "config")`. We have implemented a function that migrates previous configuration files from one location to another with a message. This message appears only once to inform you of the migration.

See Also

`tools::R_user_dir()`

Other cache utilities: `catr_clear_cache()`

Examples

```
# Caution! This modifies your current state
## Not run:
my_cache <- catr_detect_cache_dir()

# Set an example cache
ex <- file.path(tempdir(), "example", "cacheneu")
catr_set_cache_dir(ex)

catr_detect_cache_dir()

# Restore initial cache
catr_set_cache_dir(my_cache)
identical(my_cache, catr_detect_cache_dir())

## End(Not run)

catr_detect_cache_dir()
```

catr_srs_values *Reference SRS codes for* **R**[hrefhttps://CRAN.R-project.org/package=CatastRo](https://CRAN.R-project.org/package=CatastRo) **CatastRo** *services*

Description

A **tibble** including the valid SRS (also known as CRS) values that may be used in each API service. Values are provided as **EPSG codes**.

Format

A **tibble** with 16 rows and columns:

SRS Spatial Reference System (CRS) value, identified by the corresponding **EPSG** code.

Description Description of the SRS/EPSG code.

ovc_service Logical. Whether this code is valid for OVC services.

wfs_service Logical. Whether this code is valid for WFS INSPIRE services.

Details

Table: Content of [catr_srs_values](#)

SRS	Description	ovc_service	wfs_service
3785	Web Mercator	FALSE	TRUE
3857	Web Mercator	FALSE	TRUE
4230	Geográficas en ED 50	TRUE	FALSE
4258	Geográficas en ETRS89	TRUE	TRUE
4326	Geográficas en WGS 80	TRUE	TRUE
23029	UTM huso 29N en ED50	TRUE	FALSE
23030	UTM huso 30N en ED50	TRUE	FALSE
23031	UTM huso 31N en ED50	TRUE	FALSE
25829	UTM huso 29N en ETRS89	TRUE	TRUE
25830	UTM huso 30N en ETRS89	TRUE	TRUE
25831	UTM huso 31N en ETRS89	TRUE	TRUE
32627	UTM huso 27N en WGS 84	TRUE	FALSE
32628	UTM huso 28N en WGS 84	TRUE	FALSE
32629	UTM huso 29N en WGS 84	TRUE	FALSE
32630	UTM huso 30N en WGS 84	TRUE	FALSE
32631	UTM huso 31N en WGS 84	TRUE	FALSE

References

- **OVCCoordenadas**.
- **WFS INSPIRE Service**.

See Also

`sf::st_crs()`.

Other databases: `catr_atom_get_address_db_all()`, `catr_atom_get_buildings_db_all()`, `catr_atom_get_parcel_s`, `catr_atom_search_munic()`

Related WFS INSPIRE functions: `catr_wfs_get_address_bbox()`, `catr_wfs_get_buildings_bbox()`, `catr_wfs_get_parcel_s_bbox()`, `inspire_wfs_get()`

Related OVCCoordenadas functions: `catr_ovc_get_cpmrc()`, `catr_ovc_get_rccoor()`, `catr_ovc_get_rccoor_distan`

Examples

```
data("catr_srs_values")

# OVC valid codes
library(dplyr)

catr_srs_values |> filter(ovc_service)

# WFS valid codes

catr_srs_values |> filter(wfs_service)

# Use with sf::st_crs()

catr_srs_values |>
  filter(wfs_service & ovc_service) |>
  print() |>
  # Select the first value.
  slice_head(n = 1) |>
  pull(SRS) |>
  # Convert to a CRS.
  sf::st_crs(.)
```

`catr_wfs_get_address_bbox`

WFS INSPIRE: download addresses

Description

Get the spatial data of addresses. The WFS service allows performing several types of queries:

- By bounding box: `catr_wfs_get_address_bbox()` extracts objects included in the provided bounding box. See **Bounding box**.
- By street code: `catr_wfs_get_address_codvia()` extracts objects for specific addresses.
- By cadastral reference: `catr_wfs_get_address_rc()` extracts objects for specific cadastral references.
- By postal codes: `catr_wfs_get_address_postalcode()` extracts objects for specific postal codes.

Usage

```
catr_wfs_get_address_bbox(x, srs = NULL, verbose = FALSE)
```

```
catr_wfs_get_address_codvia(codvia, del, mun, srs = NULL, verbose = FALSE)
```

```
catr_wfs_get_address_rc(rc, srs = NULL, verbose = FALSE)
```

```
catr_wfs_get_address_postalcode(postalcode, srs = NULL, verbose = FALSE)
```

Arguments

x	See Bounding box . Can be one of: <ul style="list-style-type: none"> • A numeric vector of length 4 with the coordinates that define the bounding box: c(xmin, ymin, xmax, ymax). • A sf/sfc object, as provided by the sf package.
srs	SRS/CRS to use in the query. To see allowed values, use catr_srs_values , specifically the wfs_service column. See Bounding box .
verbose	Logical. If TRUE, displays informational messages.
codvia	Cadastral street code.
del	Cadastral office code.
mun	Cadastral municipality code.
rc	The cadastral reference to be extracted.
postalcode	Postal code.

Value

A [sf](#) object.

API Limits

The API service is limited to a bounding box of 4 km² and a maximum of 5,000 elements.

Bounding box

When x is a numeric vector, make sure that the srs matches the coordinate values. Additionally, the function queries the bounding box on [EPSG:25830](#) - ETRS89 / UTM zone 30N, to overcome a potential bug on the API side.

When x is a [sf](#) object, the value srs is ignored. In this case, the bounding box of the [sf](#) object is used for the query (see [sf::st_bbox\(\)](#)).

The result is always provided in the SRS of the [sf](#) object provided as input.

References

[API documentation](#).

[INSPIRE services for cadastral cartography](#).

See Also

Related INSPIRE API functions: `catr_atom_get_address()`, `catr_atom_get_address_db_all()`, `catr_atom_get_buildings()`, `catr_atom_get_buildings_db_all()`, `catr_atom_get_parcel()`, `catr_atom_get_parcel_db_all()`, `catr_wfs_get_buildings_bbox()`, `catr_wfs_get_parcel_bbox()`, `catr_wms_get_layer()`, `inspire_wfs_get()`

Related WFS INSPIRE functions: `catr_srs_values`, `catr_wfs_get_buildings_bbox()`, `catr_wfs_get_parcel_bbox()`, `inspire_wfs_get()`

Other addresses: `catr_atom_get_address()`, `catr_atom_get_address_db_all()`

Other spatial: `catr_atom_get_address()`, `catr_atom_get_buildings()`, `catr_atom_get_parcel()`, `catr_wfs_get_buildings_bbox()`, `catr_wfs_get_parcel_bbox()`, `catr_wms_get_layer()`

Examples

```
ad <- catr_wfs_get_address_bbox(  
  c(  
    233673, 4015968, 233761, 4016008  
  ),  
  srs = 25830  
)
```

```
library(ggplot2)
```

```
ggplot(ad) +  
  geom_sf()
```

`catr_wfs_get_buildings_bbox`

WFS INSPIRE: download buildings

Description

Get the spatial data of buildings. The WFS service allows performing two types of queries:

- By bounding box: `catr_wfs_get_buildings_bbox()` extracts objects included in the provided bounding box. See **Bounding box**.
- By cadastral reference: `catr_wfs_get_buildings_rc()` extracts objects for specific cadastral references.

Usage

```

catr_wfs_get_buildings_bbox(
  x,
  what = c("building", "buildingpart", "other"),
  srs = NULL,
  verbose = FALSE
)

catr_wfs_get_buildings_rc(
  rc,
  what = c("building", "buildingpart", "other"),
  srs = NULL,
  verbose = FALSE
)

```

Arguments

x	See Bounding box . Can be one of: <ul style="list-style-type: none"> • A numeric vector of length 4 with the coordinates that define the bounding box: <code>c(xmin, ymin, xmax, ymax)</code>. • A <code>sf/sfc</code> object, as provided by the <code>sf</code> package.
what	Information to load. Options are: <ul style="list-style-type: none"> • "building" for buildings. • "buildingpart" for parts of a building. • "other" for other elements such as swimming pools.
srs	SRS/CRS to use in the query. To see allowed values, use <code>catr_srs_values</code> , specifically the <code>wfs_service</code> column. See Bounding box .
verbose	Logical. If TRUE, displays informational messages.
rc	The cadastral reference to be extracted.

Value

A `sf` object.

API Limits

The API service is limited to a bounding box of 4 km² and a maximum of 5,000 elements.

Bounding box

When `x` is a numeric vector, make sure that the `srs` matches the coordinate values. Additionally, the function queries the bounding box on [EPSG:25830](#) - ETRS89 / UTM zone 30N, to overcome a potential bug on the API side.

When `x` is a `sf` object, the value `srs` is ignored. In this case, the bounding box of the `sf` object is used for the query (see `sf::st_bbox()`).

The result is always provided in the SRS of the `sf` object provided as input.

References

[API documentation.](#)

[INSPIRE services for cadastral cartography.](#)

See Also

Related INSPIRE API functions: `catr_atom_get_address()`, `catr_atom_get_address_db_all()`, `catr_atom_get_buildings()`, `catr_atom_get_buildings_db_all()`, `catr_atom_get_parcel_bbox()`, `catr_atom_get_parcel_bbox_db_all()`, `catr_wfs_get_address_bbox()`, `catr_wfs_get_parcel_bbox()`, `catr_wms_get_layer()`, `inspire_wfs_get()`

Related WFS INSPIRE functions: `catr_srs_values`, `catr_wfs_get_address_bbox()`, `catr_wfs_get_parcel_bbox()`, `inspire_wfs_get()`

Other buildings: `catr_atom_get_buildings()`, `catr_atom_get_buildings_db_all()`

Other spatial: `catr_atom_get_address()`, `catr_atom_get_buildings()`, `catr_atom_get_parcel_bbox()`, `catr_wfs_get_address_bbox()`, `catr_wfs_get_parcel_bbox()`, `catr_wms_get_layer()`

Examples

```
# Using a bbox
building <- catr_wfs_get_buildings_bbox(
  c(
    376550,
    4545424,
    376600,
    4545474
  ),
  srs = 25830
)
library(ggplot2)
ggplot(building) +
  geom_sf() +
  labs(title = "Search using bbox")

# Using a cadastral reference
rc <- catr_wfs_get_buildings_rc("6656601UL7465N")
library(ggplot2)
ggplot(rc) +
  geom_sf() +
  labs(title = "Search using rc")
```

catr_wfs_get_parcel_bbox

WFS INSPIRE: download cadastral parcels

Description

Get the spatial data of cadastral parcels and zones. The WFS service allows several types of queries:

- By bounding box: `catr_wfs_get_parcel_bbox()` extracts objects included in the provided bounding box. See **Bounding box**.
- By zoning: `catr_wfs_get_parcel_zoning()` extracts objects for a specific cadastral zone.
- By cadastral parcel: `catr_wfs_get_parcel_parcel()` extracts cadastral parcels for a specific cadastral reference.
- Neighbor cadastral parcels: `catr_wfs_get_parcel_neigh_parcel()` extracts neighbor cadastral parcels for a specific cadastral reference.
- Cadastral parcels by zoning: `catr_wfs_get_parcel_parcel_zoning()` extracts cadastral parcels for a specific cadastral zone.

Usage

```
catr_wfs_get_parcel_bbox(
  x,
  what = c("parcel", "zoning"),
  srs = NULL,
  verbose = FALSE
)
```

```
catr_wfs_get_parcel_zoning(cod_zona, srs = NULL, verbose = FALSE)
```

```
catr_wfs_get_parcel_parcel(rc, srs = NULL, verbose = FALSE)
```

```
catr_wfs_get_parcel_neigh_parcel(rc, srs = NULL, verbose = FALSE)
```

```
catr_wfs_get_parcel_parcel_zoning(cod_zona, srs = NULL, verbose = FALSE)
```

Arguments

<code>x</code>	See Bounding box . Can be one of: <ul style="list-style-type: none"> • A numeric vector of length 4 with the coordinates that define the bounding box: <code>c(xmin, ymin, xmax, ymax)</code>. • A <code>sf/sfc</code> object, as provided by the <code>sf</code> package.
<code>what</code>	Information to load. Options are: <ul style="list-style-type: none"> • "parcel" for cadastral parcels. • "zoning" for cadastral zoning.
<code>srs</code>	SRS/CRS to use in the query. To see allowed values, use <code>catr_srs_values</code> , specifically the <code>wfs_service</code> column. See Bounding box .
<code>verbose</code>	Logical. If TRUE, displays informational messages.
<code>cod_zona</code>	Cadastral zone code.
<code>rc</code>	The cadastral reference to be extracted.

Value

A `sf` object.

API Limits

The API service is limited to the following constraints:

- "parcel": Bounding box of 1 km² and a maximum of 5,000 elements.
- "zoning": Bounding box of 25 km² and a maximum of 5,000 elements.

Bounding box

When `x` is a numeric vector, make sure that the `srs` matches the coordinate values. Additionally, the function queries the bounding box on [EPSG:25830](#) - ETRS89 / UTM zone 30N, to overcome a potential bug on the API side.

When `x` is a `sf` object, the value `srs` is ignored. In this case, the bounding box of the `sf` object is used for the query (see `sf::st_bbox()`).

The result is always provided in the SRS of the `sf` object provided as input.

References

[API documentation](#).

[INSPIRE services for cadastral cartography](#).

See Also

Related INSPIRE API functions: `catr_atom_get_address()`, `catr_atom_get_address_db_all()`, `catr_atom_get_buildings()`, `catr_atom_get_buildings_db_all()`, `catr_atom_get_parcel_bbox()`, `catr_atom_get_parcel_bbox_db_all()`, `catr_wfs_get_address_bbox()`, `catr_wfs_get_buildings_bbox()`, `catr_wms_get_layer()`, `inspire_wfs_get()`

Related WFS INSPIRE functions: `catr_srs_values`, `catr_wfs_get_address_bbox()`, `catr_wfs_get_buildings_bbox()`, `inspire_wfs_get()`

Other parcels: `catr_atom_get_parcel_bbox()`, `catr_atom_get_parcel_bbox_db_all()`

Other spatial: `catr_atom_get_address()`, `catr_atom_get_buildings()`, `catr_atom_get_parcel_bbox()`, `catr_wfs_get_address_bbox()`, `catr_wfs_get_buildings_bbox()`, `catr_wms_get_layer()`

Examples

```
cp <- catr_wfs_get_parcel_bbox(  
  c(  
    233673, 4015968, 233761, 4016008  
  ),  
  srs = 25830  
)  
  
library(ggplot2)
```

```
ggplot(cp) +
  geom_sf()
```

catr_wms_get_layer *WMS INSPIRE: download map images*

Description

Get geotagged images from the Spanish Cadastre. This function is a wrapper of `mapSpain::esp_get_tiles()`.

Usage

```
catr_wms_get_layer(
  x,
  srs = NULL,
  what = c("building", "buildingpart", "parcel", "zoning", "address", "admboundary",
           "admunit"),
  styles = "default",
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE,
  crop = FALSE,
  options = NULL,
  ...
)
```

Arguments

x	See Bounding box . Can be one of: <ul style="list-style-type: none"> A numeric vector of length 4 with the coordinates that define the bounding box: <code>c(xmin, ymin, xmax, ymax)</code>. A <code>sf/sfc</code> object, as provided by the sf package.
srs	SRS/CRS to use in the query. To see allowed values, use <code>catr_srs_values</code> , specifically the <code>wfs_service</code> column. See Bounding box .
what, styles	Layer and style of the WMS layer to be downloaded. See Layers and styles .
update_cache	Logical. Should the cached file be refreshed? Defaults to FALSE. When set to TRUE, it forces a new download.
cache_dir	Path to a cache directory. On NULL, the function stores cached files in a temporary directory (see <code>base::tempdir()</code>).
verbose	Logical. If TRUE, displays informational messages.
crop	logical. If TRUE, the results will be cropped to the specified x extent. If x is an <code>sf</code> object with one POINT, crop is set to FALSE. See <code>terra::crop()</code> .
options	A named list containing additional options to pass to the query.

... Arguments passed on to `mapSpain::esp_get_tiles`

`res` character string or number. Only valid for WMS providers. Resolution (in pixels) of the final tile.

`bbox_expand` number. Expansion percentage of the bounding box of `x`.

`transparent` logical. Provides transparent background, if supported.

`mask` logical. TRUE if the result should be masked to `x`. See `terra::mask()`.

Value

A `SpatRaster` is returned, with 3 (RGB) or 4 (RGBA) layers, see `terra::RGB()`.

Bounding box

When `x` is a numeric vector, make sure that the `srs` matches the coordinate values. When `x` is a `sf` object, the value `srs` is ignored.

The query is performed using [EPSG:3857](#) (Web Mercator) and the tile is projected back to the SRS of `x`. In case that the tile looks deformed, try either providing `x` or specify the SRS of the requested tile via the `srs` argument, which should match the SRS of `x`. See **Examples**.

Layers and styles

Layers:

The argument what defines the layer to be extracted. The equivalence with the [API documentation](#) reference is:

- "parcel": CP.CadastralParcel
- "zoning": CP.CadastralZoning
- "building": BU.Building
- "buildingpart": BU.BuildingPart
- "address": AD.Address
- "admboundary": AU.AdministrativeBoundary
- "admunit": AU.AdministrativeUnit

Styles:

The WMS service provides different styles for each layer (what argument). Some available styles are:

- "parcel": Styles: "BoundariesOnly", "ReferencePointOnly", "ELFCadastre".
- "zoning": Styles: "BoundariesOnly", "ELFCadastre".
- "building", "buildingpart": "ELFCadastre".
- "address": "Number.ELFCadastre".
- "admboundary", "admunit": "ELFCadastre".

Check the [API documentation](#) for more information.

See Also

`mapSpain::esp_get_tiles()` and `terra::RGB()`. For plotting see `terra::plotRGB()` and `tidyterra::geom_spatraster`

Related INSPIRE API functions: `catr_atom_get_address()`, `catr_atom_get_address_db_all()`, `catr_atom_get_buildings()`, `catr_atom_get_buildings_db_all()`, `catr_atom_get_parcel()`, `catr_atom_get_parcel_db_all()`, `catr_wfs_get_address_bbox()`, `catr_wfs_get_buildings_bbox()`, `catr_wfs_get_parcel_bbox()`, `inspire_wfs_get()`

Other spatial: `catr_atom_get_address()`, `catr_atom_get_buildings()`, `catr_atom_get_parcel()`, `catr_wfs_get_address_bbox()`, `catr_wfs_get_buildings_bbox()`, `catr_wfs_get_parcel_bbox()`

Examples

```
# With a bbox

pict <- catr_wms_get_layer(
  c(222500, 4019500, 223700, 4020700),
  srs = 25830,
  what = "parcel"
)

library(mapSpain)
library(ggplot2)
library(tidyterra)

ggplot() +
  geom_spatraster_rgb(data = pict)

# With a spatial object

parcels <- catr_wfs_get_parcel_neigh_parcel("3662303TF3136B", srs = 25830)

# Use styles

parcels_img <- catr_wms_get_layer(parcels,
  what = "buildingpart",
  srs = 25830, # Same as the parcels object
  bbox_expand = 0.3,
  styles = "ELFCadastre"
)

ggplot() +
  geom_sf(data = parcels, fill = "blue", alpha = 0.5) +
  geom_spatraster_rgb(data = parcels_img)
```

inspire_wfs_get

Client tool for WFS INSPIRE services

Description

Access WFS INSPIRE services. This function is used internally in WFS calls and is exposed for users and developers accessing other cadastral or INSPIRE resources.

Usage

```

inspire_wfs_get(
    scheme = "https",
    hostname = "ovc.catastro.meh.es",
    path = "INSPIRE/wfsCP.aspx",
    query = list(),
    verbose = FALSE
)

```

Arguments

scheme	Character string. Protocol to access the resource on the Internet.
hostname	Character string. Host that holds the resource.
path	Character string. Specific resource in the host to access.
query	Named list. Names and values of arguments for the query.
verbose	Logical. If TRUE, displays informational messages.

Details

This function is used internally in all the WFS calls. We expose it to make it available to other users and/or developers for accessing other cadastral or INSPIRE resources. See **Examples**.

Value

Character string. Path of the resulting file in the `tempfile()` folder.

See Also

Related INSPIRE API functions: `catr_atom_get_address()`, `catr_atom_get_address_db_all()`, `catr_atom_get_buildings()`, `catr_atom_get_buildings_db_all()`, `catr_atom_get_parcel()`, `catr_atom_get_parcel_db_all()`, `catr_wfs_get_address_bbox()`, `catr_wfs_get_buildings_bbox()`, `catr_wfs_get_parcel_bbox()`, `catr_wms_get_layer()`

Related WFS INSPIRE functions: `catr_srs_values`, `catr_wfs_get_address_bbox()`, `catr_wfs_get_buildings_bbox()`, `catr_wfs_get_parcel_bbox()`

Examples

```
# Access the Cadastre of Navarra
# Try also https://ropenspain.github.io/CatastroNav/

file_local <- inspire_wfs_get(
  hostname = "inspire.navarra.es",
  path = "services/BU/wfs",
  query = list(
    service = "WFS",
    request = "getfeature",
    typenames = "BU:Building",
    bbox = "609800,4740100,611000,4741300",
    SRSNAME = "EPSG:25830"
  )
)

if (!is.null(file_local)) {
  pamp <- sf::read_sf(file_local)

  library(ggplot2)
  ggplot(pamp) +
    geom_sf()
}
```

Index

* ATOM

- catr_atom_get_address, 2
- catr_atom_get_address_db_all, 4
- catr_atom_get_buildings, 6
- catr_atom_get_buildings_db_all, 7
- catr_atom_get_parcel, 9
- catr_atom_get_parcel_db_all, 11
- catr_atom_search_munic, 12

* INSPIRE

- catr_atom_get_address, 2
- catr_atom_get_address_db_all, 4
- catr_atom_get_buildings, 6
- catr_atom_get_buildings_db_all, 7
- catr_atom_get_parcel, 9
- catr_atom_get_parcel_db_all, 11
- catr_wfs_get_address_bbox, 26
- catr_wfs_get_buildings_bbox, 28
- catr_wfs_get_parcel_bbox, 30
- catr_wms_get_layer, 33
- inspire_wfs_get, 36

* OVCCallejero

- catr_ovc_get_cod_munic, 16
- catr_ovc_get_cod_provinces, 18

* OVCCoordenadas

- catr_ovc_get_cpmrc, 19
- catr_ovc_get_rccoor, 20
- catr_ovc_get_rccoor_distancia, 21
- catr_srs_values, 25

* WFS

- catr_srs_values, 25
- catr_wfs_get_address_bbox, 26
- catr_wfs_get_buildings_bbox, 28
- catr_wfs_get_parcel_bbox, 30
- inspire_wfs_get, 36

* WMS

- catr_wms_get_layer, 33

* addresses

- catr_atom_get_address, 2
- catr_atom_get_address_db_all, 4

- catr_wfs_get_address_bbox, 26

* buildings

- catr_atom_get_buildings, 6
- catr_atom_get_buildings_db_all, 7
- catr_wfs_get_buildings_bbox, 28

* cache utilities

- catr_clear_cache, 13
- catr_set_cache_dir, 23

* cadastral references

- catr_ovc_get_cpmrc, 19
- catr_ovc_get_rccoor, 20
- catr_ovc_get_rccoor_distancia, 21

* databases

- catr_atom_get_address_db_all, 4
- catr_atom_get_buildings_db_all, 7
- catr_atom_get_parcel_db_all, 11
- catr_atom_search_munic, 12
- catr_srs_values, 25

* parcels

- catr_atom_get_parcel, 9
- catr_atom_get_parcel_db_all, 11
- catr_wfs_get_parcel_bbox, 30

* search

- catr_atom_search_munic, 12
- catr_get_code_from_coords, 14
- catr_ovc_get_cod_munic, 16
- catr_ovc_get_cod_provinces, 18

* spatial

- catr_atom_get_address, 2
- catr_atom_get_buildings, 6
- catr_atom_get_parcel, 9
- catr_wfs_get_address_bbox, 26
- catr_wfs_get_buildings_bbox, 28
- catr_wfs_get_parcel_bbox, 30
- catr_wms_get_layer, 33

base::grep(), 5, 8, 11

base::tempdir(), 3, 5, 6, 8, 10, 11, 13, 15,
23, 33

- catr_atom_get_address, 2
- catr_atom_get_address(), 5, 7–10, 12, 13, 28, 30, 32, 35, 36
- catr_atom_get_address_db_all, 4
- catr_atom_get_address_db_all(), 3, 7–10, 12, 13, 26, 28, 30, 32, 35, 36
- catr_atom_get_address_db_to (catr_atom_get_address_db_all), 4
- catr_atom_get_buildings, 6
- catr_atom_get_buildings(), 3, 5, 8–10, 12, 13, 28, 30, 32, 35, 36
- catr_atom_get_buildings_db_all, 7
- catr_atom_get_buildings_db_all(), 3, 5, 7, 10, 12, 13, 26, 28, 30, 32, 35, 36
- catr_atom_get_buildings_db_to (catr_atom_get_buildings_db_all), 7
- catr_atom_get_parcel, 9
- catr_atom_get_parcel(), 3, 5, 7–9, 12, 13, 28, 30, 32, 35, 36
- catr_atom_get_parcel_db_all, 11
- catr_atom_get_parcel_db_all(), 3, 5, 7–10, 13, 26, 28, 30, 32, 35, 36
- catr_atom_get_parcel_db_to (catr_atom_get_parcel_db_all), 11
- catr_atom_search_munic, 12
- catr_atom_search_munic(), 3, 5–7, 9, 10, 12, 13, 16–18, 26
- catr_clear_cache, 13
- catr_clear_cache(), 24
- catr_detect_cache_dir (catr_set_cache_dir), 23
- catr_detect_cache_dir(), 23, 24
- catr_get_code_from_coords, 14
- catr_get_code_from_coords(), 13, 17, 18
- catr_ovc_get_cod_munic, 16
- catr_ovc_get_cod_munic(), 13, 14, 16, 18
- catr_ovc_get_cod_provinces, 18
- catr_ovc_get_cod_provinces(), 13, 16, 17
- catr_ovc_get_cpmrc, 19
- catr_ovc_get_cpmrc(), 21, 22, 26
- catr_ovc_get_rccoor, 20
- catr_ovc_get_rccoor(), 20, 22, 26
- catr_ovc_get_rccoor_distancia, 21
- catr_ovc_get_rccoor_distancia(), 20, 21, 26
- catr_set_cache_dir, 23
- catr_set_cache_dir(), 14
- catr_srs_values, 5, 9, 12, 13, 15, 19–22, 25, 25, 27–33, 36
- catr_wfs_get_address_bbox, 26
- catr_wfs_get_address_bbox(), 3, 5, 7, 8, 10, 12, 26, 30, 32, 35, 36
- catr_wfs_get_address_codvia (catr_wfs_get_address_bbox), 26
- catr_wfs_get_address_postalcode (catr_wfs_get_address_bbox), 26
- catr_wfs_get_address_rc (catr_wfs_get_address_bbox), 26
- catr_wfs_get_buildings_bbox, 28
- catr_wfs_get_buildings_bbox(), 3, 5, 7–10, 12, 26, 28, 32, 35, 36
- catr_wfs_get_buildings_rc (catr_wfs_get_buildings_bbox), 28
- catr_wfs_get_parcel_bbox, 30
- catr_wfs_get_parcel_bbox(), 3, 5, 7, 8, 10, 12, 26, 28, 30, 35, 36
- catr_wfs_get_parcel_neigh_parcel (catr_wfs_get_parcel_bbox), 30
- catr_wfs_get_parcel_parcel (catr_wfs_get_parcel_bbox), 30
- catr_wfs_get_parcel_parcel_zoning (catr_wfs_get_parcel_bbox), 30
- catr_wfs_get_parcel_zoning (catr_wfs_get_parcel_bbox), 30
- catr_wms_get_layer, 33
- catr_wms_get_layer(), 3, 5, 7, 8, 10, 12, 28, 30, 32, 36
- esp_dict_region_code(), 15
- inspire_wfs_get, 36
- inspire_wfs_get(), 3, 5, 7, 8, 10, 12, 26, 28, 30, 32, 35
- mapSpain::esp_get_munic_siane, 15
- mapSpain::esp_get_munic_siane(), 16, 17
- mapSpain::esp_get_tiles, 34
- mapSpain::esp_get_tiles(), 33, 35
- regex, 15
- sf, 3, 6, 10, 14, 15, 27, 29, 32–34
- sf::st_bbox(), 27, 29, 32

`sf::st_centroid()`, 16
`sf::st_crs()`, 26
`SpatRaster`, 34

`tempdir()`, 24
`tempfile()`, 36
`terra::crop()`, 33
`terra::mask()`, 34
`terra::plotRGB()`, 35
`terra::RGB()`, 34, 35
`tibble`, 5, 8, 11, 13, 15–19, 21, 22, 25
`tidyterra::geom_spatraster_rgb()`, 35
`tools::R_user_dir()`, 14, 24