Package: arcgisplaces (via r-universe)

October 31, 2026

Title Search for POIs using ArcGIS 'Places Service'

Version 0.1.2

Description The ArcGIS 'Places service' is a ready-to-use location service that can search for businesses and geographic locations around the world. It allows you to find, locate, and discover detailed information about each place. Query for places near a point, within a bounding box, filter based on categories, or provide search text. 'arcgisplaces' integrates with 'sf' for out of the box compatibility with other spatial libraries. Learn more in the 'Places service' API reference <https://developers.arcgis.com/rest/places/>.

License Apache License (>= 2)

Encoding UTF-8

Language en

RoxygenNote 7.3.2

Imports arcgisutils (>= 0.3.0), cli, httr2 (>= 1.0.5), rlang, wk

Config/rextendr/version 0.3.1.9001

SystemRequirements Cargo (Rust's package manager), rustc, OpenSSL

Suggests sf

Depends R (>= 4.2)

LazyData true

NeedsCompilation yes

Author Josiah Parry [aut, cre] (<https://orcid.org/0000-0001-9910-865X>)

Maintainer Josiah Parry <josiah.parry@gmail.com>

Config/pak/sysreqs libssl-dev libclang-dev

Repository https://cran.r-universe.dev

Date/Publication 2025-04-10 16:20:02 UTC

RemoteUrl https://github.com/cran/arcgisplaces

RemoteRef HEAD

RemoteSha 819f587813c125567017d59045e8d759f815764e

Contents

categories 2
category_details 3
fields 4
near_point 5
place_details 6
within_extent 8

Index **10**

`categories` *Return the name and category ID of all categories, or categories which satisfy a filter*

Description

A category describes a type of place, such as "movie theater" or "zoo". The places service has over 1,000 categories (or types) of place. The categories fall into ten general groups: Arts and Entertainment, Business and Professional Services, Community and Government, Dining and Drinking, Events, Health and Medicine, Landmarks and Outdoors, Retail, Sports and Recreation, and Travel and Transportation.

Usage

```
categories(
  search_text = NULL,
  icon = NULL,
  language = NULL,
  token = arc_token()
)
```

Arguments

`search_text`	Default `NULL`. Free search text for places against names, categories etc. Must be a scalar value.
`icon`	Default `NULL`. Must be one of `"svg"`, `"png"` `"cim"`. Determines whether icons are returned and the type of icon to use with a place or category.
`language`	Optional case-sensitive parameter to specify the preferred language to.
`token`	an object of class `httr2_token` as generated by `auth_code()` or related function

Details

The categories are organized into a hierarchical system where a general category contains many more detailed variations on the parent category. For example: "Travel and Transportation" (Level 1), "Transport Hub" (Level 2), "Airport" (Level 3) and "Airport Terminal" (Level 4). The hierarchy has up to 5 levels of categories.

Value

A `data.frame` with columns:

- `category_id`: the unique identifier for the category
- `full_label`: a list of character vectors containing all labels for the category
- `icon_url`: a character vector containing the icon URL if present
- `parents`: a list of character vectors containing the parent `category_id` values

References

API Documentation

Examples

```
## Not run:
categories("Coffee Shop")

## End(Not run)
```

`category_details` *Get the category details for a category ID.*

Description

The `/categories/{categoryId}` request returns all the groups to which the category belongs. You must supply a category ID to use this request. Note: Query parameters are case-sensitive.

Usage

```
category_details(
  category_id,
  icon = NULL,
  language = NULL,
  token = arc_token(),
  .progress = TRUE
)
```

Arguments

`category_id`	Default `NULL`. A character vector which filters places to those that match the category IDs.
`icon`	Default `NULL`. Must be one of `"svg"`, `"png"` `"cim"`. Determines whether icons are returned and the type of icon to use with a place or category.
`language`	Optional case-sensitive parameter to specify the preferred language to.
`token`	an object of class `httr2_token` as generated by `auth_code()` or related function
`.progress`	Default `TRUE`. Whether a progress bar should be provided.

Details

Language Codes:

The language codes use the CLDR (Common Locale Data Repository) format string that uses a two letter language code (e.g. "fr" for French) optionally followed by a two letter country code (e.g. "fr-CA" for French in Canada).

If an unsupported language code is used, then the service will attempt to fall-back to the closest available language. This is done by stripping regional and extension subtags to find a known language code. For example, French Canadian (fr-CA) is unsupported so this falls back to French fr.

Should the fallback fail, then the service will return category names in the default language en for English.

Value

A `data.frame` with columns:

- `category_id`
- `full_label`: a list of character vectors
- `icon_url`: a character vector of the URL to an icon, if available
- `parents`: a list of character vectors indicating the parent category ID

References

API Documentation

Examples

```
## Not run:
categories <- c(
  "12015", "11172", "15015", "19027", "13309", "16069", "19004",
  "13131", "18046", "15048"
)
category_details(categories)

## End(Not run)
```

`fields` *Possible Fields to Return from Place Details*

Description

A character vector containing the possible return fields that define the attributes to return. Use in `place_details()`'s `requested_fields` argument.

Usage

```
fields
```

Format

An object of class `character` of length 40.

Source

https://developers.arcgis.com/rest/places/place-id-get/#requestedfields

---

`near_point` *Search for places near a point by radius*

Description

Finds places that are within a given radius of a specified location. The returned places contain basic data such as name, category and location.

Usage

```
near_point(
  x,
  y,
  radius = 1000,
  search_text = NULL,
  category_id = NULL,
  icon = NULL,
  token = arc_token()
)
```

Arguments

`x`	The x coordinate, or longitude, to search from, in WGS84 decimal degrees.
`y`	The y coordinate, or latitude, to search from, in WGS84 decimal degrees.
`radius`	Default `1000`. The radius in meters to search for places. Maximum value of `10000`.
`search_text`	Default `NULL`. Free search text for places against names, categories etc. Must be a scalar value.
`category_id`	Default `NULL`. A character vector which filters places to those that match the category IDs.
`icon`	Default `NULL`. Must be one of `"svg"`, `"png"` `"cim"`. Determines whether icons are returned and the type of icon to use with a place or category.
`token`	an object of class `httr2_token` as generated by `auth_code()` or related function

Value

An `sf` object with columns

- `place_id`: The unique Id of this place. The ID can be passed to `place_details()` to retrieve additional details.
- `name`: The name of the place, or point of interest. You can search for places by name using the searchText property
- `distance`: A double vector of the distance, in meters, from the place to the search point.
- `categories`: A `data.frame` with two columns `category_id` and `label`. Categories are uniquely identified by a categoryId. For example, `17119` identifies a "Bicycle Store" and `10051` identifies a "Stadium". Note that a single place can belong to multiple categories (for example, a petrol station could also have a super-market).
- `icon`: A character vector of the URL for an icon for this place or category in either svg, cim or png format.
- `geometry`: an `sfc_POINT` object in `EPSG:4326`

References

API Documentation

Examples

```
## Not run:
near_point(-117.194769, 34.057289)
near_point(139.75, 35.66)

## End(Not run)
```

`place_details` *Get place details including name, address, description, and other attributes*

Description

The `/places/{placeId}` request returns details for a place.

Usage

```
place_details(
  place_id,
  requested_fields,
  icon = NULL,
  token = arc_token(),
  .progress = TRUE
)
```

Arguments

`place_id`	a character vector of place IDs as generated by `within_extent()` or `near_point()`.
`requested_fields`	Required. See API Reference for possible fields or refer to the `fields` vector.
`icon`	Default `NULL`. Must be one of `"svg"`, `"png"` `"cim"`. Determines whether icons are returned and the type of icon to use with a place or category.
`token`	an object of class `httr2_token` as generated by `auth_code()` or related function
`.progress`	Default `TRUE`. Whether a progress bar should be provided.

Details

To request details, you use the `requested_fields` argument to specify the fields and the attributes you want from the Place, Address, Details and/or Location price groups.

It is always recommended to specify the fields you want, however, you can also use `requested_fields=all` to return all of the attributes available. By default, The `place_id` attribute is always returned in addition to the other attributes you requested.

The attributes available for places may vary. For example, opening hours may not be available (or applicable) for geographic places or landmarks.

You will only be charged for attributes that contain valid values for the requested fields. If no data is available for the requested field, null or an empty collection is returned and you are not charged. You are only charged once if one or more attributes with valid values are returned from a price group.

Value

an sf object

References

API Documentation

Examples

```
## Not run:
place_ids <- c(
  "37f1062ae1c3d37511003e382b08ca32",
  "9cdd210841deedef0e3309bdd3fe47f1"
)

res <- place_details(place_ids)

## End(Not run)
```

`within_extent` *Search for places within an extent (bounding box).*

Description

The `/places/within-extent` request searches for places within an extent (bounding box).

Usage

```
within_extent(
  xmin,
  ymin,
  xmax,
  ymax,
  search_text = NULL,
  category_id = NULL,
  icon = NULL,
  token = arc_token()
)
```

Arguments

`xmin`	The minimum x coordinate, or longitude, of the search extent in `EPSG:4326`.
`ymin`	The minimum y coordinate, or latitude, of the search extent in `EPSG:4326`
`xmax`	The maximum x coordinate, or longitude, of the search extent in `EPSG:4326`.
`ymax`	The maximum y coordinate, or latitude, of the search extent in `EPSG:4326`
`search_text`	Default `NULL`. Free search text for places against names, categories etc. Must be a scalar value.
`category_id`	Default `NULL`. A character vector which filters places to those that match the category IDs.
`icon`	Default `NULL`. Must be one of `"svg"`, `"png"` `"cim"`. Determines whether icons are returned and the type of icon to use with a place or category.
`token`	an object of class `httr2_token` as generated by `auth_code()` or related function

Details

You must supply the `xmin`, `ymin`, `xmax` and `ymax` coordinates to define the extent. The maximum width and height of an extent that can be used in a search is 20,000 meters.

You can also provide multiple categories or search text to find specific types of places within the extent.

Note: You cannot permanently store places. Please see the Terms of use.

Note: Query parameters are case-sensitive.

Value

An sf object with columns

- place_id: The unique Id of this place. The ID can be passed to place_details() to retrieve additional details.
- name: The name of the place, or point of interest. You can search for places by name using the searchText property
- categories: A data.frame with two columns category_id and label. Categories are uniquely identified by a categoryId. For example, 17119 identifies a "Bicycle Store" and 10051 identifies a "Stadium". Note that a single place can belong to multiple categories (for example, a petrol station could also have a super-market).
- icon: A character vector of the URL for an icon for this place or category in either svg, cim or png format.
- geometry: an sfc_POINT object in EPSG:4326

References

API Documentation

Examples

```
## Not run:
within_extent(
  139.74,
  35.65,
  139.75,
  35.66,
  category_ids = "10001"
)

## End(Not run)
```

Index

∗ **datasets**
fields, 4

auth_code(), *2*, *3*, *5*, *7*, *8*

categories, 2
category_details, 3

fields, 4, *7*

near_point, 5

place_details, 6
place_details(), *4*

within_extent, 8