Package 'rstac'

February 1, 2023

Title Client Library for SpatioTemporal Asset Catalog

Version 0.9.2-2

Description Provides functions to access, search and download spacetime earth observation data via SpatioTemporal Asset Catalog (STAC). This package supports the version 1.0.0 (and older) of the STAC specification (<https://github.com/radiantearth/stac-spec>).
For further details see Simoes et al. (2021) <doi:10.1109/IGARSS47720.2021.9553518>.

License MIT + file LICENSE

URL https://brazil-data-cube.github.io/rstac/

BugReports https://github.com/brazil-data-cube/rstac/issues

Encoding UTF-8

RoxygenNote 7.2.3

Depends R (>= 3.5)

Imports httr, crayon, utils, jsonlite, lifecycle, magrittr

Suggests testthat, sf, knitr, rmarkdown, png, jpeg, tibble, dplyr, purrr, slider, leaflet, tmap, stars, ggplot2, geojsonsf

Collate 'cql2-expr-funs.R' 'cql2-types.R' 'parse-utils.R' 'cql2-core.R' 'cql2-json.R' 'cql2-text.R' 'cql2-adv_comp.R' 'cql2-funs.R' 'cql2-env.R' 'cql2-utils.R' 'assets-utils.R' 'assets-funs.R' 'check-utils.R' 'conformance-query.R' 'collections-query.R' 'deprec-funs.R' 'document-funs.R' 'ext_filter.R' 'ext_query.R' 'extensions.R' 'items-funs.R' 'items-utils.R' 'items-query.R' 'message-utils.R' 'preview-utils.R' 'print.R' 'query-funs.R' 'queryables-query.R' 'request.R' 'signatures.R' 'stac-query.R' 'stac_search.R' 'stac_version.R' 'url-utils.R' 'utils.R' 'rstac.R'

VignetteBuilder knitr

NeedsCompilation no

Author Rolf Simoes [aut],
Felipe Carvalho [aut, cre],
Brazil Data Cube Team [aut],
National Institute for Space Research (INPE) [cph]

Maintainer Felipe Carvalho <lipecaso@gmail.com>

Repository CRAN

Date/Publication 2023-02-01 18:00:02 UTC

R topics documented:

`assets_filter` *Assets filter (Deprecated)*

Description

[Deprecated]

Usage

```
assets_filter(items, ..., filter_fn = NULL)

## S3 method for class 'STACItemCollection'
assets_filter(items, ..., filter_fn = NULL)

## S3 method for class 'STACItem'
assets_filter(items, ..., filter_fn = NULL)
```

Arguments

items	a STACItemCollection object representing the result of /stac/search, /collections/{collectionId
...	additional arguments. See details.
filter_fn	a function that will be used to filter the attributes listed in the properties.

Value

a list with the attributes of date, bands and paths.

---

assets_functions *Assets functions*

Description

These functions provide support to work with STACItemCollection and STACItem item objects.

- assets_download(): Downloads the assets provided by the STAC API.
- assets_url(): **[Experimental]** Returns a character vector with each asset href. For the URL, you can add the GDAL library drivers for the following schemes: HTTP/HTTPS files, S3 (AWS S3) and GS (Google Cloud Storage).
- assets_select(): **[Experimental]** Selects the assets of each item by its name (asset_names parameter), by expressions (... parameter), or by a selection function (select_fn parameter). Note: This function can produce items with empty assets. In this case, users can use the items_compact() function to remove items with no assets.
- assets_rename(): **[Experimental]** Rename each asset using a named list or a function.

Usage

```
assets_download(
  items,
  asset_names = NULL,
  output_dir = getwd(),
  overwrite = FALSE,
  ...,
  download_fn = NULL,
  fn = deprecated()
)

## S3 method for class 'STACItem'
assets_download(
  items,
  asset_names = NULL,
  output_dir = getwd(),
  overwrite = FALSE,
  ...,
```

```
  create_json = FALSE,
  download_fn = NULL,
  fn = deprecated()
)

## S3 method for class 'STACItemCollection'
assets_download(
  items,
  asset_names = NULL,
  output_dir = getwd(),
  overwrite = FALSE,
  ...,
  download_fn = NULL,
  create_json = TRUE,
  items_max = Inf,
  progress = TRUE,
  fn = deprecated()
)

## Default S3 method:
assets_download(
  items,
  asset_names = NULL,
  output_dir = getwd(),
  overwrite = FALSE,
  ...,
  create_json = FALSE,
  download_fn = NULL,
  fn = deprecated()
)

assets_url(items, asset_names = NULL, append_gdalvsi = FALSE)

## S3 method for class 'STACItem'
assets_url(items, asset_names = NULL, append_gdalvsi = FALSE)

## S3 method for class 'STACItemCollection'
assets_url(items, asset_names = NULL, append_gdalvsi = FALSE)

## Default S3 method:
assets_url(items, asset_names = NULL, append_gdalvsi = FALSE)

assets_select(items, ..., asset_names = NULL, select_fn = NULL)

## S3 method for class 'STACItem'
assets_select(items, ..., asset_names = NULL, select_fn = NULL)

## S3 method for class 'STACItemCollection'
```

```
assets_select(items, ..., asset_names = NULL, select_fn = NULL)

## Default S3 method:
assets_select(items, ..., asset_names = NULL, select_fn = NULL)

assets_rename(items, mapper = NULL, ...)

## S3 method for class 'STACItem'
assets_rename(items, mapper = NULL, ...)

## S3 method for class 'STACItemCollection'
assets_rename(items, mapper = NULL, ...)

## Default S3 method:
assets_rename(items, mapper = NULL, ...)

has_assets(items)

## S3 method for class 'STACItem'
has_assets(items)

## S3 method for class 'STACItemCollection'
has_assets(items)

## Default S3 method:
has_assets(items)

asset_key()

asset_eo_bands(field)

asset_raster_bands(field)
```

Arguments

items	a STACItem or STACItemCollection object representing the result of /stac/search, /collections/{collectionId}/items or /collections/{collectionId}/items/{itemId} endpoints.
asset_names	a character vector with the names of the assets to be selected.
output_dir	a character directory in which the assets will be saved. Default is the working directory (getwd())
overwrite	a logical if TRUE will replace the existing file, if FALSE, a warning message is shown.
...	additional arguments. See details.
download_fn	a function to handle download of assets for each item to be downloaded. Using this function, you can change the hrefs for each asset, as well as the way download is done.

fn	**[Deprecated]** use download_fn parameter instead.
create_json	a logical indicating if a JSON file with item metadata (STACItem or STACItemCollection) must be created in the output directory.
items_max	a numeric corresponding to how many items will be downloaded.
progress	a logical indicating if a progress bar must be shown or not. Defaults to TRUE.
append_gdalvsi	a logical value. If true, gdal drivers are included in the URL of each asset. The following schemes are supported: HTTP/HTTPS files, S3 (AWS S3) and GS (Google Cloud Storage).
select_fn	a function to select assets an item (STACItem or STACItemCollection). This function receives as parameter the asset element and, optionally, the asset name. Asset elements contain metadata describing spatial-temporal objects. Users can provide a function to select assets based on this metadata by returning a logical value where TRUE selects the asset, and FALSE discards it.
mapper	either a named list or a function to rename assets of an item (STACItem or STACItemCollection). In the case of a named list, use <old name> = <new name> to rename the assets. The function can be used to rename the assets by returning a character string using the metadata contained in the asset object.
field	a character with the name of the asset field to return.

Details

Ellipsis argument (. . .) appears in different assets functions and has distinct purposes:

- assets_download(): ellipsis is used to pass additional httr options to GET or POST methods, such as add_headers or set_cookies.
- assets_select(): ellipsis is used to pass expressions that will be evaluated against each asset metadata. Expressions must be evaluated as a logical value where TRUE selects the asset and FALSE discards it. Multiple expressions are combine with AND operator. Expressions can use asset helper functions (i.e. asset_key(), asset_eo_bands(), and asset_raster_bands()). Multiple expressions are combined with AND operator.

  WARNING: Errors in the evaluation of expressions are considered as FALSE.
- assets_rename(): ellipsis is used to pass named parameters to be processed in the same way as the named list in mapper argument.

Value

- assets_download(): returns the same input object item (STACItem or STACItemCollection) where href properties point to the download assets.
- assets_url(): returns a character vector with all assets href of an item (STACItem or STACItemCollection).
- assets_select(): returns the same input object item (STACItem or STACItemCollection) with the selected assets.
- assets_rename(): returns the same input object item (STACItemCollection or STACItem) with the assets renamed.

See Also

stac_search(), items(), get_request()

Examples

```
## Not run:
 # assets_download function
 stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
   stac_search(collections = "CB4_64_16D_STK-1",
               datetime = "2019-06-01/2019-08-01") %>%
   stac_search() %>%
   get_request() %>%
   assets_download(asset_names = "thumbnail", output_dir = tempdir())

## End(Not run)

## Not run:
 # assets_url function
 stac_item <- stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
  stac_search(collections = "CB4_64_16D_STK-1", limit = 100,
         datetime = "2017-08-01/2018-03-01",
         bbox = c(-48.206,-14.195,-45.067,-12.272)) %>%
  get_request() %>% items_fetch(progress = FALSE)

 stac_item %>% assets_url()

## End(Not run)

## Not run:
 # assets_select function
 stac_item <- stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
  stac_search(collections = "CB4_64_16D_STK-1", limit = 100,
         datetime = "2017-08-01/2018-03-01",
         bbox = c(-48.206,-14.195,-45.067,-12.272)) %>%
  get_request() %>% items_fetch(progress = FALSE)

 stac_item %>% assets_select(asset_names = "NDVI")

## End(Not run)

## Not run:
items <- stac("https://planetarycomputer.microsoft.com/api/stac/v1") %>%
  stac_search(collections = c("landsat-8-c2-l2", "sentinel-2-l2a"),
              bbox = c(xmin = -64.85976089, ymin = -10.49199395,
                       xmax = -64.79272527, ymax =-10.44736091),
              datetime = "2019-01-01/2019-06-28",
              limit = 50) %>%
  post_request()

# Selects assets by name
items <- assets_select(items,
                       asset_names = c("B02", "B03", "SR_B1", "SR_B2"))
```

```
# Renames the landsat assets
items <- assets_rename(items,
                       SR_B1 = "blue",
                       SR_B2 = "green",
                       B02   = "blue",
                       B03   = "green")
# Get the assets url's
assets_url(items)

## End(Not run)
```

`collections` *Endpoint functions*

Description

The `collections` function implements the WFS3 `/collections` and `/collections/{collectionId}` endpoints.

Each endpoint retrieves specific STAC objects:

- `/collections`: Returns a list of STAC Collections published in the STAC service
- `/collections/{collectionId}`: Returns a single STAC Collection object

Usage

```
collections(q, collection_id = NULL)
```

Arguments

`q`	a `RSTACQuery` object expressing a STAC query criteria.
`collection_id`	a `character` collection id to be retrieved.

Value

A `RSTACQuery` object with the subclass `collections` for `/collections/` endpoint, or a `collection_id` subclass for `/collections/{collection_id}` endpoint, containing all search field parameters to be provided to STAC API web service.

See Also

`get_request()`, `post_request()`, `items()`

Examples

```
## Not run:
 stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
   collections() %>%
   get_request()

 stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
   collections(collection_id = "CB4_64_16D_STK-1") %>%
   get_request()

## End(Not run)
```

`conformance` *Conformance endpoint*

Description

The conformance endpoint provides the capabilities of the service. This endpoint is accessible from the provider's catalog (`/conformance`).

Usage

```
conformance(q)
```

Arguments

`q`	a `RSTACQuery` object expressing a STAC query criteria.

Value

A `RSTACQuery` object with the subclass `conformance` for `/conformance` endpoint.

See Also

`get_request()`, `stac()`, `collections()`

Examples

```
## Not run:
stac("https://planetarycomputer.microsoft.com/api/stac/v1") %>%
  conformance() %>% get_request()

## End(Not run)
```

cql2_helpers *CQL2 helper function*

Description

These are helper functions to easy construction CQL2 expressions. These functions are not meant to be used in expressions and they must be escaped using {{ to be evaluated before request.

Usage

```
cql2_bbox_as_geojson(bbox)

cql2_date(x)

cql2_timestamp(x)

cql2_interval(start = "..", end = "..")
```

Arguments

bbox	a numeric containing a bbox with c(xmin, ymin, xmax, ymax).
x, start, end	a character string containing valid date or timestamp.

Details

- cql2_bbox_as_geojson(): used to convert bounding box (bbox) to a GeoJSON object to be used as argument of CQL2 spatial operators.
- cql2_date(), cql2_timestamp(), and cql2_interval(): create temporal literal values to be passed into CQL2 expressions.

Value

- cql2_bbox_as_geojson(): GeoJSON object.
- cql2_date(), cql2_timestamp(), and cql2_interval(): internal rstac expressions representing temporal values.

Examples

```
## Not run:
bbox <- c(-122.2751, 47.5469, -121.9613, 47.7458)

cql2_json(
    collection == "landsat-c2-l2" &&
      t_intersects(datetime, {{
        cql2_interval("2020-12-01", "2020-12-31")
      }}) &&
      s_intersects(geometry, {{
```

```
        cql2_bbox_as_geojson(bbox)
      }})
  )

## End(Not run)
```

`ext_filter` *Filter extension*

Description

[Experimental] `ext_filter()` implements Common Query Language (CQL2) filter extension on `rstac`. This extension expands the filter capabilities providing a query language to construct more complex expressions. CQL2 is an OGC standard and defines how filters can be constructed. It supports predicates for standard data types like strings, numbers, and boolean as well as for spatial geometries (point, lines, polygons) and temporal data (instants and intervals).

[Experimental] `cql2_json()` and `cql2_text()` are helper functions that can be used to show how expressions are converted into CQL2 standard, either JSON or TEXT formats.

`rstac` translates R expressions to CQL2, allowing users to express their filter criteria using R language. For more details on how to create CQL2 expressions in `rstac`. See the details section.

Usage

```
ext_filter(q, expr, lang = NULL, crs = NULL)

cql2_json(expr)

cql2_text(expr)
```

Arguments

`q`	a `RSTACQuery` object expressing a STAC query criteria.
`expr`	a valid R expression to be translated to CQL2 (see details).
`lang`	a character value indicating which CQL2 representation to be used. It can be either `"cql2-text"` (for plain text) or `"cql2-json"` (for JSON format). If `NULL` (default), `"cql2-text"` is used for HTTP `GET` requests and `"cql2-json"` for `POST` requests.
`crs`	an optional character value informing the coordinate reference system used by geometry objects. If `NULL` (default), STAC services assume `"WGS 84"`.

Details

To allow users to express filter criteria in R language, `rstac` takes advantage of the abstract syntax tree (AST) to translate R expressions to CQL2 expressions. The following topics describe the correspondences between `rstac` expressions and CQL2 operators.

Standard comparison operators:

- ==, >=, <=, >, <, and != operators correspond to =, >=, <=, >, <, and <> in CQL2, respectively.
- function `is_null(a)` and `!is_null(a)` corresponds to `a IS NULL` and `a IS NOT NULL` CQL2 operators, respectively.

Advanced comparison operators:

- `a %like% b` corresponds to CQL2 `a LIKE b`, `a` and `b` `strings` values.
- `between(a, b, c)` corresponds to CQL2 `a BETWEEN b AND c`, where `b` and `c` `integer` values.
- `a %in% b` corresponds to CQL2 `a IN (b)`, where `b` should be a list of values of the same type as `a`.

Spatial operators:

- functions `s_intersects(a, b)`, `s_touches(a, b)`, `s_within(a, b)`, `s_overlaps(a, b)`, `s_crosses(a, b)`, and `s_contains(a, b)` corresponds to CQL2 `S_INTERSECTS(a, b)`, `S_TOUCHES(a, b)`, `S_WITHIN(a, b)`, `S_OVERLAPS(a, b)`, `S_CROSSES(a, b)`, and `S_CONTAINS(a, b)` operators, respectively. Here, `a` and `b` should be `geometry` objects. `rstac` accepts `sf`, `sfc`, `sfg`, `list` (representing GeoJSON objects), or `character` (representing either GeoJSON or WKT).
- **NOTE**: All of the above spatial object types, except for the `character`, representing a WKT, may lose precision due to numeric truncation when R converts numbers to JSON text. WKT strings are sent "as is" to the service. Therefore, the only way for users to retain precision on spatial objects is to represent them as a WKT string. However, user can control numeric precision using the `options(stac_digits = ...)`. The default value is 15 digits.

Temporal operators:

- functions `date(a)`, `timestamp(a)`, and `interval(a, b)` corresponds to CQL2 `DATE(a)`, `TIMESTAMP(a)`, and `INTERVAL(a, b)` operators, respectively. These functions create literal `temporal` values. The first two define an `instant` type, and the third an `interval` type.
- functions `t_after(a, b)`, `t_before(a, b)`, `t_contains(a, b)`, `t_disjoint(a, b)`, `t_during(a, b)`, `t_equals(a, b)`, `t_finishedby(a, b)`, `t_finishes(a, b)`, `t_intersects(a, b)`, `t_meets(a, b)`, `t_meet(a, b)`, `t_metby(a, b)`, `t_overlappedby(a, b)`, `t_overlaps(a, b)`, `t_startedby(a, b)`, and `t_starts(a, b)` corresponds to CQL2 `T_AFTER(a, b)`, `T_BEFORE(a, b)`, `T_CONTAINS(a, b)`, `T_DISJOINT(a, b)`, `T_DURING(a, b)`, `T_EQUALS(a, b)`, `T_FINISHEDBY(a, b)`, `T_FINISHES(a, b)`, `T_INTERSECTS(a, b)`, `T_MEETS(a, b)`, `T_MEET(a, b)`, `T_METBY(a, b)`, `T_OVERLAPPEDBY(a, b)`, `T_OVERLAPS(a, b)`, `T_STARTEDBY(a, b)`, and `T_STARTS(a, b)` operators, respectively. Here, `a` and `b` are `temporal` values (`instant` or `interval`, depending on function).

Array Operators:

- R unnamed lists (or vectors of size > 1) are translated to arrays by `rstac`. `list()` and `c()` functions always create `array` values in CQL2 context, no matter the number of its arguments.
- functions `a_equals(a, b)`, `a_contains(a, b)`, `a_containedby(a, b)`, and `a_overlaps(a, b)` corresponds to CQL2 `A_EQUALS(a, b)`, `A_CONTAINS(a, b)`, `A_CONTAINEDBY(a, b)`, and `A_OVERLAPS(a, b)` operators, respectively. Here, `a` and `b` should be `arrays`.

Value

A RSTACQuery object with the subclass ext_filter containing all request parameters to be passed to get_request() or post_request() function.

Note

The specification states that double-quoted identifiers should be interpreted as properties. However, the R language does not distinguish double quote from single quote strings. The right way to represent double quoted properties in R is to use the escape character (), for example "date"‘.

See Also

ext_query(), stac_search(), post_request(), endpoint(), before_request(), after_response(), content_response()

Examples

```
## Not run:
# Standard comparison operators in rstac:
# Creating a stac search query
req <- rstac::stac("https://planetarycomputer.microsoft.com/api/stac/v1") %>%
  rstac::stac_search(limit = 5)

# Equal operator '=' with collection property
req %>% ext_filter(collection == "sentinel-2-l2a") %>% post_request()

# Not equal operator '!=' with collection property
req %>% ext_filter(collection != "sentinel-2-l2a") %>% post_request()

# Less than operator '<' with cloud_cover property
req %>% ext_filter(`eo:cloud_cover` < 10) %>% post_request()

# Greater than operator '>' with vegetation_percentage property
req %>% ext_filter(`s2:vegetation_percentage` > 50) %>% post_request()

# Less than or equal operator '<=' with datetime property
req %>% ext_filter(datetime <= "1986-01-01") %>% post_request()

# Greater than or equal '>=' with AND operator
req %>% ext_filter(collection == "sentinel-2-l2a"   &&
                   `s2:vegetation_percentage` >= 50 &&
                   `eo:cloud_cover` <= 10) %>% post_request()
# Advanced comparison operators
# 'LIKE' operator
req %>% ext_filter(collection %like% "modis%") %>% post_request()

# 'IN' operator
req %>% ext_filter(
  collection %in% c("landsat-c2-l2", "sentinel-2-l2a") &&
    datetime > "2019-01-01" &&
    datetime < "2019-06-01") %>%
```

```
  post_request()

# Spatial operator
# Lets create a polygon with list
polygon <- list(
  type = "Polygon",
  coordinates = list(
    matrix(c(-62.34499836, -8.57414572,
             -62.18858174, -8.57414572,
             -62.18858174, -8.15351185,
             -62.34499836, -8.15351185,
             -62.34499836, -8.57414572),
           ncol = 2, byrow = TRUE)
  )
)
# 'S_INTERSECTS' spatial operator with polygon and geometry property
req %>% ext_filter(collection == "sentinel-2-l2a" &&
                   s_intersects(geometry, {{polygon}})) %>% post_request()

# 'S_CONTAINS' spatial operator with point and geometry property
point <- list(type = "Point", coordinates = c(-62.45792211, -8.61158488))
req %>% ext_filter(collection == "landsat-c2-l2" &&
                   s_contains(geometry, {{point}})) %>% post_request()

# 'S_CROSSES' spatial operator with linestring and geometry property
linestring <- list(
  type = "LineString",
  coordinates = matrix(
         c(-62.55735320, -8.43329465, -62.21791603, -8.36815014),
         ncol = 2, byrow = TRUE
  )
)
req %>% ext_filter(collection == "landsat-c2-l2" &&
                   s_crosses(geometry, {{linestring}})) %>% post_request()

# Temporal operator
# 'T_INTERSECTS' temporal operator with datetime property
req %>% ext_filter(
  collection == "landsat-c2-l2" &&
    t_intersects(datetime, interval("1985-07-16T05:32:00Z",
                                    "1985-07-24T16:50:35Z"))) %>%
 post_request()

# 'T_DURING' temporal operator with datetime property
req %>%
 ext_filter(collection == "landsat-c2-l2" &&
            t_during(datetime,
            interval("2022-07-16T05:32:00Z", ".."))) %>%
 post_request()

# 'T_BEFORE' temporal operator with datetime property
req %>%
 ext_filter(collection == "landsat-c2-l2" &&
```

```
            t_before(datetime, timestamp("2022-07-16T05:32:00Z"))) %>%
 post_request()

# 'T_AFTER' temporal operator with datetime property
req %>%
 ext_filter(collection == "landsat-c2-l2" &&
            t_after(datetime, timestamp("2022-07-16T05:32:00Z"))) %>%
  post_request()

# Shows how CQL2 expression (TEXT format)
cql2_text(collection == "landsat-c2-l2" &&
  s_crosses(geometry, {{linestring}}))

# Shows how CQL2 expression (JSON format)
cql2_json(collection == "landsat-c2-l2" &&
            t_after(datetime, timestamp("2022-07-16T05:32:00Z")))

## End(Not run)
```

`ext_query` *Query extension*

Description

The `ext_query()` is the *exported function* of the STAC API query extension. It can be used after a call to `stac_search()` function. It allows that additional fields and operators other than those defined in `stac_search()` function be used to make a complex filter.

The function accepts multiple filter criteria. Each filter entry is an expression formed by `<field>` `<operator>` `<value>`, where `<field>` refers to a valid item property. Supported `<fields>` depends on STAC API service implementation. The users must rely on the service providers' documentation to know which properties can be used by this extension.

The `ext_query()` function allows the following `<operators>`

- == corresponds to '`eq`'
- != corresponds to '`neq`'
- < corresponds to '`lt`'
- <= corresponds to '`lte`'
- > corresponds to '`gt`'
- >= corresponds to '`gte`'
- `\%startsWith\%` corresponds to '`startsWith`' and implements a string prefix search operator.
- `\%endsWith\%` corresponds to '`endsWith`' and implements a string suffix search operator.
- `\%contains\%`: corresponds to '`contains`' and implements a string infix search operator.
- `\%in\%`: corresponds to '`in`' and implements a vector search operator.

Besides this function, the following S3 generic methods were implemented to get things done for this extension:

- The `endpoint()` for subclass `ext_query`
- The `before_request()` for subclass `ext_query`
- The `after_response()` for subclass `ext_query`

See source file `ext_query.R` for an example of how to implement new extensions.

Usage

```
ext_query(q, ...)
```

Arguments

`q`	a `RSTACQuery` object expressing a STAC query criteria.
`...`	entries with format `<field> <operator> <value>`.

Value

A `RSTACQuery` object with the subclass `ext_query` containing all request parameters to be passed to `post_request()` function.

See Also

`ext_filter()`, `stac_search()`, `post_request()`, `endpoint()`, `before_request()`, `after_response()`, `content_response()`

Examples

```
## Not run:
 stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
   stac_search(collections = "CB4_64_16D_STK-1") %>%
   ext_query("bdc:tile" %in% "022024") %>%
   post_request()

## End(Not run)
```

---

get_request *STAC API request functions*

Description

The `get_request` is function that makes HTTP GET requests to STAC web services, retrieves, and parse the data.

The `post_request` is function that makes HTTP POST requests to STAC web services, retrieves, and parse the data.

Usage

```
get_request(q, ...)

post_request(q, ..., encode = c("json", "multipart", "form"))
```

Arguments

q	a RSTACQuery object expressing a STAC query criteria.
...	config parameters to be passed to GET or POST methods, such as add_headers or set_cookies.
encode	a character informing the request body Content-Type. Accepted types are 'json' ('application/json'), 'form' ('application/x-www-form-urlencoded'), and 'multipart' ('multipart/form-data'). Defaults to 'json'.

Value

Either a STACCatalog, STACCollection, STACCollectionList, STACItemCollection or STACItem object depending on the subclass and search fields parameters of q argument.

See Also

stac() stac_search() collections() items()

Examples

```
## Not run:
 stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
  get_request()

 stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
  stac_search(collections = "CB4_64_16D_STK-1") %>%
  post_request()

## End(Not run)
```

---

items *Endpoint functions*

Description

The items function implements WFS3 /collections/{collectionId}/items, and /collections/{collectionId}/ite endpoints.

Each endpoint retrieves specific STAC objects:

- /collections/{collectionId}/items: Returns a STAC Items collection (GeoJSON)
- /collections/{collectionId}/items/{itemId}: Returns a STAC Item (GeoJSON Feature)

The endpoint /collections/{collectionId}/items accepts the same filters parameters of stac_search() function.

Usage

```
items(q, feature_id = NULL, datetime = NULL, bbox = NULL, limit = NULL)
```

Arguments

q	a RSTACQuery object expressing a STAC query criteria.
feature_id	a character with item id to be fetched. Only works if the collection_id is informed. This is equivalent to the endpoint /collections/{collectionId}/items/{featureId}.
datetime	a character with a date-time or an interval. Date and time strings needs to conform to RFC 3339. Intervals are expressed by separating two date-time strings by '/' character. Open intervals are expressed by using '..' in place of date-time. Examples: • A date-time: "2018-02-12T23:20:50Z" • A closed interval: "2018-02-12T00:00:00Z/2018-03-18T12:31:12Z" • Open intervals: "2018-02-12T00:00:00Z/.." or "../2018-03-18T12:31:12Z" Only features that have a datetime property that intersects the interval or datetime informed in datetime are selected.
bbox	a numeric vector with only features that have a geometry that intersects the bounding box are selected. The bounding box is provided as four or six numbers, depending on whether the coordinate reference system includes a vertical axis (elevation or depth): • Lower left corner, coordinate axis 1 • Lower left corner, coordinate axis 2 • Lower left corner, coordinate axis 3 (optional) • Upper right corner, coordinate axis 1 • Upper right corner, coordinate axis 2 • Upper right corner, coordinate axis 3 (optional) The coordinate reference system of the values is WGS84 longitude/latitude (http://www.opengis.net/def/crs/OGC/1.3/CRS84). The values are, in most cases, the sequence of minimum longitude, minimum latitude, maximum longitude, and maximum latitude. However, in cases where the box spans the antimeridian, the first value (west-most box edge) is larger than the third value (east-most box edge).
limit	an integer defining the maximum number of results to return. If not informed, it defaults to the service implementation.

Value

A RSTACQuery object with the subclass items for /collections/{collection_id}/items endpoint, or a item_id subclass for /collections/{collection_id}/items/{feature_id} endpoint, containing all search field parameters to be provided to STAC API web service.

See Also

get_request(), post_request(), collections()

Examples

```
## Not run:
 stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
   collections("CB4_64_16D_STK-1") %>%
   items(bbox = c(-47.02148, -17.35063, -42.53906, -12.98314)) %>%
   get_request()

 stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
   collections("CB4_64_16D_STK-1") %>%
   items("CB4_64_16D_STK_v001_022023_2020-07-11_2020-07-26") %>%
   get_request()

## End(Not run)
```

`items_group` *Items functions*

Description

These functions provide support to work with `STACItemCollection` and `STACItem` objects.

- `items_length()`: shows how many items there are in the `STACItemCollection` object.
- `items_matched()`: shows how many items matched the search criteria. It supports `search:metadata` (v0.8.0), `context` (v0.9.0), and `numberMatched` (OGC WFS3 core spec).
- `items_fetch()`: request all STAC Items through pagination.
- `items_next()`: fetches a new page from STAC service.
- `items_datetime()`: retrieves the `datetime` field in `properties` from `STACItemCollection` and `STACItem` objects.
- `items_bbox()`: retrieves the `bbox` field of a `STACItemCollection` or a `STACItem` object.
- `item_assets()`: returns the assets name from `STACItemCollection` and `STACItem` objects.
- `items_filter()`: selects only items that match some criteria (see details section).
- `items_reap()`: extract key values by traversing all items in a `STACItemCollection` object.
- `items_fields()`: lists field names inside an item.
- `items_group()`: **[Deprecated]** organizes items as elements of a list using some criteria.
- `items_sign()`: allow access assets by preparing its url.
- `items_as_sf()`: **[Experimental]** convert items to `sf` object.

Usage

```
items_group(items, ..., field = NULL, index = NULL)

items_length(items)

## S3 method for class 'STACItem'
items_length(items)

## S3 method for class 'STACItemCollection'
items_length(items)

## Default S3 method:
items_length(items)

items_matched(items, matched_field = NULL)

## S3 method for class 'STACItem'
items_matched(items, matched_field = NULL)

## S3 method for class 'STACItemCollection'
items_matched(items, matched_field = NULL)

## Default S3 method:
items_matched(items, matched_field = NULL)

items_fetch(items, ...)

## S3 method for class 'STACItemCollection'
items_fetch(items, ..., progress = TRUE, matched_field = NULL)

items_next(items, ...)

## S3 method for class 'STACItemCollection'
items_next(items, ...)

items_datetime(items)

## S3 method for class 'STACItem'
items_datetime(items)

## S3 method for class 'STACItemCollection'
items_datetime(items)

## Default S3 method:
items_datetime(items)

items_bbox(items)
```

```
## S3 method for class 'STACItem'
items_bbox(items)

## S3 method for class 'STACItemCollection'
items_bbox(items)

## Default S3 method:
items_bbox(items)

items_assets(items, simplify = deprecated())

## S3 method for class 'STACItem'
items_assets(items, simplify = deprecated())

## S3 method for class 'STACItemCollection'
items_assets(items, simplify = deprecated())

## Default S3 method:
items_assets(items, simplify = deprecated())

items_filter(items, ..., filter_fn = NULL)

## S3 method for class 'STACItemCollection'
items_filter(items, ..., filter_fn = NULL)

items_compact(items)

## S3 method for class 'STACItemCollection'
items_compact(items)

items_reap(items, field, ..., pick_fn = identity)

## S3 method for class 'STACItem'
items_reap(items, field, ..., pick_fn = identity)

## S3 method for class 'STACItemCollection'
items_reap(items, field, ..., pick_fn = identity)

## Default S3 method:
items_reap(items, field, ..., pick_fn = identity)

items_fields(items, field = NULL, ...)

## S3 method for class 'STACItem'
items_fields(items, field = NULL, ...)

## S3 method for class 'STACItemCollection'
items_fields(items, field = NULL, ...)
```

```
## Default S3 method:
items_fields(items, field = NULL, ...)

items_sign(items, sign_fn)

## S3 method for class 'STACItem'
items_sign(items, sign_fn)

## S3 method for class 'STACItemCollection'
items_sign(items, sign_fn)

## Default S3 method:
items_sign(items, sign_fn)

items_as_sf(items)

## S3 method for class 'STACItem'
items_as_sf(items)

## S3 method for class 'STACItemCollection'
items_as_sf(items)
```

Arguments

items	a STACItemCollection object.
...	additional arguments. See details.
field	a character with the names of the field to get the subfields values.
index	an atomic vector with values as the group index.
matched_field	a character vector with the path where the number of items returned in the named list is located starting from the initial node of the list. For example, if the information is in a position items$meta$found of the object, it must be passed as the following parameter c("meta", "found").
progress	a logical indicating if a progress bar must be shown or not. Defaults to TRUE.
simplify	**[Deprecated]** no side-effect
filter_fn	a function that receives an item that should evaluate a logical value.
pick_fn	a function used to pick elements from items addressed by field parameter.
sign_fn	a function that receives an item as a parameter and returns an item signed.

Details

Ellipsis argument (...) appears in different items functions and has distinct purposes:

- items_matched() and items_assets(): ellipsis is not used.
- items_fetch() and items_next(): ellipsis is used to pass additional httr options to GET or POST methods, such as add_headers or set_cookies.

- `items_fields()`: ellipsis parameter is deprecated in version 0.9.2 of rstac. Please, use `field` parameter instead.
- `items_filter()`: ellipsis is used to pass logical expressions to be evaluated against a `STACItem` field as filter criteria.

  WARNING: the evaluation of filter expressions changed in `rstac` 0.9.2. Older versions of `rstac` used `properties` field to evaluate filter expressions. Below, there is an example of how to write expressions in new `rstac` version:

  ```
  # expression in older version
  items_filter(stac_obj, `eo:cloud_cover` < 10)
  # now expressions must refer to properties explicitly
  items_filter(stac_obj, properties$`eo:cloud_cover` < 10)
  items_filter(stac_obj, properties[["eo:cloud_cover"]] < 10)
  ```

- `items_sign()`: in the near future, ellipsis will be used to append key-value pairs to the url query string of an asset.

`items_sign()` has `sign_fn` parameter that must be a function that receives as argument an item and returns a signed item. `rstac` provides `sign_bdc()` and `sign_planetary_computer()` functions to access Brazil Data Cube products and Microsoft Planetary Computer catalogs, respectively.

Value

- `items_length()`: an `integer` value.
- `items_matched()`: returns an `integer` value if the STAC web server does support this extension. Otherwise returns `NULL`.
- `items_fetch()`: a `STACItemCollection` with all matched items.
- `items_next()`: fetches a new page from STAC service.
- `items_datetime()`: a `list` of all items' datetime.
- `items_bbox()`: returns a `list` with all items' bounding boxes.
- `item_assets()`: Returns a `character` value with all assets names of the all items.
- `items_filter()`: a `STACItemCollection` object.
- `items_reap()`: a `vector` if the supplied field is atomic, otherwise or a `list`.
- `items_fields()`: a `character` vector.
- `items_group()`: a `list` of `STACItemCollection` objects.
- `items_sign()`: a `STACItemCollection` object with signed assets url.
- `items_as_sf()`: a `sf` object.

Examples

```
## Not run:
 x <- stac("https://brazildatacube.dpi.inpe.br/stac") %>%
     stac_search(collections = "CB4_64_16D_STK-1") %>%
     stac_search(limit = 500) %>%
     get_request()
```

```
 x %>% items_length()
 x %>% items_matched()
 x %>% items_datetime()
 x %>% items_bbox()
 x %>% items_fetch()

## End(Not run)

## Not run:
# Defining BDC token
Sys.setenv("BDC_ACCESS_KEY" = "token-123")

# STACItem object
stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
    stac_search(collections = "CB4_64_16D_STK-1", limit = 100,
        datetime = "2017-08-01/2018-03-01",
        bbox = c(-48.206, -14.195, -45.067, -12.272)) %>%
    get_request() %>% items_sign(sign_fn = sign_bdc())


## End(Not run)

## Not run:
# STACItemCollection object
stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
    stac_search(collections = "CB4_64_16D_STK-1", limit = 100,
        datetime = "2017-08-01/2018-03-01",
        bbox = c(-48.206, -14.195, -45.067, -12.272)) %>%
    get_request() %>%
    items_filter(properties$`eo:cloud_cover` < 10)

# Example with AWS STAC
stac("https://earth-search.aws.element84.com/v0") %>%
  stac_search(collections = "sentinel-s2-l2a-cogs",
              bbox = c(-48.206, -14.195, -45.067, -12.272),
              datetime = "2018-06-01/2018-06-30",
              limit = 500) %>%
  post_request() %>%
  items_filter(filter_fn = function(x) {x$properties$`eo:cloud_cover` < 10})

## End(Not run)

## Not run:
# STACItemCollection object
stac_item <- stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
 stac_search(collections = "CB4_64_16D_STK-1", limit = 100,
        datetime = "2017-08-01/2018-03-01",
        bbox = c(-48.206, -14.195, -45.067, -12.272)) %>%
 get_request() %>% items_fetch(progress = FALSE)

stac_item %>% items_reap(field = c("properties", "datetime"))

## End(Not run)
```

preview_plot *Plot preview images*

Description

This is a helper function to plot preview assets (e.g. quicklook, thumbnail, rendered_preview). Currently, only png and jpeg formats are supported.

Usage

```
preview_plot(url)
```

Arguments

`url`	image URL to be plotted.

Value

A rastergrob grob from package `grid`.

print *Printing functions*

Description

The print function covers all objects in the rstac package:

- `stac()`: returns a `STACCatalog` document from `/stac` (v0.8.0) or `/` (v0.9.0 or v1.0.0) endpoint.
- `stac_search()`: returns a `STACItemCollection` document from `/stac/search` (v0.8.0) or `/search` (v0.9.0 or v1.0.0) endpoint containing all Items that match the provided search predicates.
- `collections()`: implements the `/collections` and `/collections/{collectionId}` endpoints. The former returns a `STACCollectionList` document that lists all collections published by the server, and the later returns a single `STACCollection` document that describes a unique collection.
- `items()`: retrieves a `STACItemCollection` document from `/collections/{collectionId}/items` endpoint and a `STACItem` document from `/collections/{collectionId}/items/{itemId}` endpoints.

The rstac package objects visualization is based on markdown, a lightweight markup language. You can paste the output into any markdown editor for a better visualization.

Call `print()` function to print the rstac's objects. You can determine how many items will be printed using `n` parameter.

Usage

```
## S3 method for class 'RSTACQuery'
print(x, ...)

## S3 method for class 'STACCatalog'
print(x, ...)

## S3 method for class 'STACCollectionList'
print(x, n = 10, ...)

## S3 method for class 'STACCollection'
print(x, ...)

## S3 method for class 'STACItemCollection'
print(x, n = 10, ..., tail = FALSE)

## S3 method for class 'STACItem'
print(x, ...)

## S3 method for class 'Queryables'
print(x, n = 10, ...)

## S3 method for class 'Conformance'
print(x, n = 5, ...)
```

Arguments

x	either a RSTACQuery object expressing a STAC query criteria or any RSTACDocument.
...	other parameters passed in the functions.
n	number of entries to print. Each object has its own rule of truncation: the STACCollection objects will print 10 links by default. If the object has less than 20 collections, all collections will be shown. In STACItemCollection, 10 features will be printed by default. To show all entries, use n = Inf.
tail	A logical value indicating if last features in STACItemCollection object must be show.

See Also

stac() stac_search() collections() items()

Examples

```
## Not run:
 # STACItemCollection object
 stac_item_collection <-
   stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
   stac_search(collections = "CB4_64_16D_STK-1",
          bbox = c(-47.02148, -17.35063, -42.53906, -12.98314),
```

```
          limit = 15) %>%
  get_request()

print(stac_item_collection, n = 10)

# STACCollectionList object
stac_collection <-
    stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
    collections() %>%
    get_request()

print(stac_collection, n = 5)

# RSTACQuery object
obj_rstac <- stac("https://brazildatacube.dpi.inpe.br/stac/")

print(obj_rstac)

## End(Not run)
```

---

queryables *Endpoint functions*

Description

The queryables endpoint allows the user to discover which properties can be used in the filter extension. This endpoint can be accessed from the catalog (`/queryables`) or from a collection (`/collections/{collection_id}/queryables`).

Usage

```
queryables(q)
```

Arguments

q	a `RSTACQuery` object expressing a STAC query criteria.

Value

A `RSTACQuery` object with the subclass `queryables` for `/queryables` endpoint.

See Also

`ext_filter()`, `conformance()`, `collections()`

Examples

```
## Not run:
# Catalog's queryables
rstac::stac("https://planetarycomputer.microsoft.com/api/stac/v1") %>%
  rstac::queryables() %>% rstac::get_request()

# Collection's queryables
rstac::stac("https://planetarycomputer.microsoft.com/api/stac/v1") %>%
  rstac::collections(collection_id = "sentinel-2-l2a") %>%
  rstac::queryables() %>%
  rstac::get_request()

## End(Not run)
```

`rstac` *R client library for STAC (rstac)*

Description

Provides functions to access, search and download spacetime earth observation data via SpatioTemporal Asset Catalog (STAC). This package supports the version 1.0.0 (and older) of the STAC specification (https://github.com/radiantearth/stac-spec). For further details see Simoes et al. (2021) doi:10.1109/IGARSS47720.2021.9553518.

The `rstac` functions

The rstac package provides two categories of functions: API endpoints and data access and organization.

STAC API endpoints functions

- `stac()`: implements STAC `/stac` endpoint for version 0.8.1 or below, and `/` for versions 0.9.0 or higher.
- `conformance()`: implements `/conformance` endpoint.
- `collections()`: implements `/collections` and `/collections/{collectionId}` endpoints.
- `items()`: implements `/collections/{collectionId}/items` and `/collections/{collectionId}/items/{featu` endpoints.
- `queryables()`: implements `/queryables` and `/collections/{collectionId}/queryables` endpoints.
- `stac_search()`: implements STAC `/stac/search` endpoint for version 0.8.1 or below, and `/search` endpoint for versions 0.9.0 or higher.
- `ext_filter()`: implements `/filter` CQL2 endpoint.

Data access and organization functions

- `get_request()`: makes HTTP GET requests to STAC web service.
- `post_request()`: makes HTTP POST requests to STAC web service.
- `items_matched()`: returns how many items matched the search criteria.
- `items_fetch()`: fetches all matched items from service.
- `items_filter()`: selects items according to some criteria.
- `items_as_sf()`: converts items to a `sf` object.
- `items_fields()`: help explore fields inside items.
- `items_compact()`: removes all items with empty assets.
- `items_reap()`: extracts contents from items.
- `items_length()`: informs how many items are fetched locally.
- `items_sign()`: appends tokens to assets' URL and turn them accessible.
- `assets_select()`: select assets in items.
- `assets_rename()`: rename assets in items.
- `assets_url()`: extract all URL to assets in items.
- `assets_download()`: download assets in batch.

Data types

The package implements the following S3 classes: `STACItemCollection`, `STACItem`, `STACCatalog`, `STACCollectionList` and `STACCollection`. These classes are regular lists representing the corresponding JSON STAC objects.

Author(s)

Maintainer: Felipe Carvalho <lipecaso@gmail.com>

Authors:

- Rolf Simoes <rolfsimoes@gmail.com>
- Brazil Data Cube Team <brazildatacube@inpe.br>

Other contributors:

- National Institute for Space Research (INPE) [copyright holder]

See Also

Useful links:

- https://brazil-data-cube.github.io/rstac/
- Report bugs at https://github.com/brazil-data-cube/rstac/issues

sign_bdc

Signature in hrefs provided by the STAC from the Brazil Data Cube project.

Description

To sign the hrefs with your token you need to store it in an environment variable in `BDC_ACCESS_KEY`or use `acess_token` parameter.

Usage

```
sign_bdc(access_token = NULL, ...)
```

Arguments

`access_token`	a `character` with the access token parameter to access Brazil Data Cube assets.
`...`	additional parameters can be supplied to the `GET` function of the `httr` package.

Value

a `function` that signs each item assets.

Examples

```
## Not run:
 # STACItemCollection object
 stac_obj <- stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
   stac_search(collections = "CB4_64_16D_STK-1",
               datetime = "2019-06-01/2019-08-01") %>%
   stac_search() %>%
   get_request()

 # signing each item href
 stac_obj %>% items_sign(sign_fn = sign_bdc(access_token = "123"))

## End(Not run)
```

sign_planetary_computer

Signature in hrefs provided by the STAC from Microsoft's Planetary Computer.

Description

To perform the signing of the hrefs a request is sent to Planetary Computer servers and the returned content corresponds to the token that will be used in the href.

Usage

```
sign_planetary_computer(..., headers = NULL, token_url = NULL)
```

Arguments

...	additional parameters can be supplied to the GET function of the httr package.
headers	a named character vector with headers key-value content.
token_url	a character with the URL that generates the tokens in the Microsoft service. By default is used: "https://planetarycomputer.microsoft.com/api/sas/v1/token"

Value

a function that signs each item assets.

Examples

```
## Not run:
 # STACItemCollection object
 stac_obj <- stac("https://planetarycomputer.microsoft.com/api/stac/v1/") %>%
  stac_search(collections = "sentinel-2-l2a",
              bbox = c(-47.02148, -17.35063, -42.53906, -12.98314)) %>%
  get_request()

 # signing each asset href
 stac_obj %>% items_sign(sign_fn = sign_planetary_computer())

 # example of access to collections that require authentication
 stac_obj <- stac("https://planetarycomputer.microsoft.com/api/stac/v1") %>%
   stac_search(collections = c("sentinel-1-rtc"),
               bbox = c(-64.8597, -10.4919, -64.79272527, -10.4473),
               datetime = "2019-01-01/2019-01-28") %>%
   post_request()

 # signing each asset href
 # stac_obj %>% items_sign(
 #   sign_fn = sign_planetary_computer(
 #     headers = c("Ocp-Apim-Subscription-Key" = <your-mpc-token>)
 #   )
 # )

## End(Not run)
```

---

stac *Endpoint functions*

Description

The stac function implements /stac API endpoint (>=0.8.0), and / for versions 0.9.0 or higher. It prepares search field parameters to be provided to a STAC API web service. This endpoint should return a STAC Catalog document containing all published data catalogs.

Usage

```
stac(base_url, force_version = NULL)
```

Arguments

base_url	a character informing the base URL of a STAC web service.
force_version	a character providing the version of the STAC used. If not provided, the rstac package will make requests to try to find the version of STAC used. It is highly recommended that you inform the STAC version you are using.

Value

A RSTACQuery object with the subclass stac containing all request parameters to be provided to API service.

See Also

stac_search(), collections(), items(), get_request(), post_request()

Examples

```
## Not run:
 stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
   get_request()

## End(Not run)
```

---

stac_search *Endpoint functions*

Description

(This document is based on STAC specification documentation https://github.com/radiantearth/stac-spec/ and reproduces some of its parts)

The stac_search function implements /stac/search API endpoint (v0.8.1) and /search (v0.9.0 or v1.0.0). It prepares query parameters used in the search API request, a stac object with all filter parameters to be provided to get_request or post_request functions. The GeoJSON content returned by these requests is a STACItemCollection object, a regular R list representing a STAC Item Collection document.

Usage

```
stac_search(
  q,
  collections = NULL,
  ids = NULL,
  bbox = NULL,
  datetime = NULL,
  intersects = NULL,
  limit = NULL
)
```

Arguments

q	a RSTACQuery object expressing a STAC query criteria.
collections	a character vector of collection IDs to include in the search for items. Only items in one of the provided collections will be searched.
ids	a character vector with item IDs. All other filters parameters that further restrict the number of search results are ignored.
bbox	a numeric vector with only features that have a geometry that intersects the bounding box are selected. The bounding box is provided as four or six numbers, depending on whether the coordinate reference system includes a vertical axis (elevation or depth): • Lower left corner, coordinate axis 1 • Lower left corner, coordinate axis 2 • Lower left corner, coordinate axis 3 (optional) • Upper right corner, coordinate axis 1 • Upper right corner, coordinate axis 2 • Upper right corner, coordinate axis 3 (optional) The coordinate reference system of the values is WGS84 longitude/latitude (http://www.opengis.net/def/crs/OGC/1.3/CRS84). The values are, in most cases, the sequence of minimum longitude, minimum latitude, maximum longitude, and maximum latitude. However, in cases where the box spans the antimeridian, the first value (west-most box edge) is larger than the third value (east-most box edge).
datetime	a character with a date-time or an interval. Date and time strings needs to conform to RFC 3339. Intervals are expressed by separating two date-time strings by '/' character. Open intervals are expressed by using '..' in place of date-time. Examples: • A date-time: "2018-02-12T23:20:50Z" • A closed interval: "2018-02-12T00:00:00Z/2018-03-18T12:31:12Z" • Open intervals: "2018-02-12T00:00:00Z/.." or "../2018-03-18T12:31:12Z" Only features that have a datetime property that intersects the interval or datetime informed in datetime are selected.

intersects	a list expressing GeoJSON geometries objects as specified in RFC 7946. Only returns items that intersect with the provided geometry. To turn a GeoJSON into a list the packages geojsonsf or jsonlite can be used.
limit	an integer defining the maximum number of results to return. If not informed, it defaults to the service implementation.

Value

A RSTACQuery object with the subclass search containing all search field parameters to be provided to STAC API web service.

See Also

stac(), ext_query(), get_request(), post_request()

Examples

```
## Not run:
 # GET request
 stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
  stac_search(collections = "CB4_64_16D_STK-1", limit = 10,
         datetime = "2017-08-01/2018-03-01") %>%
  get_request()

 # POST request
 stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
  stac_search(collections = "CB4_64_16D_STK-1",
         bbox = c(-47.02148, -17.35063, -42.53906, -12.98314)) %>%
  post_request()

## End(Not run)
```

---

stac_version *Utility functions*

Description

These function retrieves information about either rstac queries (RSTACQuery objects) or rstac documents (RSTACDocument objects).

Usage

```
stac_version(x, ...)
```

Arguments

x	either a RSTACQuery object expressing a STAC query criteria or any RSTACDocument.
...	config parameters to be passed to GET method, such as add_headers or set_cookies.

Value

The `stac_version()` function returns a `character` STAC API version.

Index