# Package 'bcdata'

November 18, 2026

**Title** Search and Retrieve Data from the BC Data Catalogue

**Version** 0.5.4

**Description** Search, query, and download tabular and 'geospatial' data from the British Columbia Data Catalogue (<https://catalogue.data.gov.bc.ca/>). Search catalogue data records based on keywords, data licence, sector, data format, and B.C. government organization. View metadata directly in R, download many data formats, and query 'geospatial' data available via the B.C. government Web Feature Service ('WFS') using 'dplyr' syntax.

**License** Apache License (== 2.0)

**URL** https://bcgov.github.io/bcdata/, https://catalogue.data.gov.bc.ca, https://github.com/bcgov/bcdata

**BugReports** https://github.com/bcgov/bcdata/issues

**Imports** methods, utils, stats, cli (>= 3.3.0), DBI (>= 1.1.0), crul (>= 1.1.0), dbplyr (>= 2.3.4), dplyr (>= 1.1.2), tibble (>= 3.1.0), glue (>= 1.6.0), jsonlite (>= 1.6.0), leaflet (>= 2.1.0), purrr (>= 0.3), readr (>= 2.1), readxl (>= 1.4.0), rlang (>= 1.0), sf (>= 1.0), tidyselect (>= 1.1.0), xml2 (>= 1.3.0)

**Suggests** covr, ggplot2, knitr, rmarkdown, testthat (>= 3.0.0), withr

**VignetteBuilder** knitr

**Encoding** UTF-8

**Collate** 'bcdata-package.R' 'bcdc-get-citation.R' 'bcdc-web-services.R' 'bcdc_browse.R' 'bcdc_options.R' 'bcdc_search.R' 'cli.R' 'cql-geom-predicates.R' 'cql-translator.R' 'describe-feature.R' 'get_data.R' 'utils-as_tibble.R' 'utils-classes.R' 'utils-collect.R' 'utils-filter.R' 'utils-is.R' 'utils-mutate.R' 'utils-pipe.R' 'utils-select.R' 'utils-show-query.R' 'utils.R' 'zzz.R'

**Config/testthat/edition** 3

**Config/testthat/parallel** true

**Config/roxygen2/version** 8.0.0

**NeedsCompilation** no

**Author** Andy Teucher [aut, cre] (ORCID:
<https://orcid.org/0000-0002-7840-692X>),
Sam Albers [aut, ctb] (ORCID: <https://orcid.org/0000-0002-9270-7884>),
Stephanie Hazlitt [aut, ctb] (ORCID:
<https://orcid.org/0000-0002-3161-2304>),
Province of British Columbia [cph]

**Maintainer** Andy Teucher <andy.teucher@gmail.com>

**Repository** CRAN

**Date/Publication** 2026-11-18 00:40:02 UTC

# Contents

bcdc_browse . . . . . 2
bcdc_check_geom_size . . . . . 4
bcdc_describe_feature . . . . . 5
bcdc_get_citation . . . . . 6
bcdc_get_data . . . . . 7
bcdc_get_record . . . . . 9
bcdc_list . . . . . 10
bcdc_list_groups . . . . . 11
bcdc_list_organizations . . . . . 11
bcdc_options . . . . . 12
bcdc_preview . . . . . 13
bcdc_query_geodata . . . . . 14
bcdc_read_functions . . . . . 16
bcdc_search . . . . . 17
bcdc_search_facets . . . . . 18
bcdc_tidy_resources . . . . . 19
CQL . . . . . 20
cql_geom_predicates . . . . . 20

**Index** **22**

---

`bcdc_browse` *Load the B.C. Data Catalogue URL into an HTML browser*

---

## Description

This is a wrapper around utils::browseURL with the URL for the B.C. Data Catalogue as the default

## Usage

```
bcdc_browse(
  query = NULL,
  browser = getOption("browser"),
  encodeIfNeeded = FALSE
)
```

## Arguments

| | |
|---|---|
| query | Default (NULL) opens a browser to https://catalogue.data.gov.bc.ca. This argument will also accept a B.C. Data Catalogue record ID or name to take you directly to that page. If the provided ID or name doesn't lead to a valid webpage, bcdc_browse will search the data catalogue for that string. |
| browser | a non-empty character string giving the name of the program to be used as the HTML browser. It should be in the PATH, or a full path specified. Alternatively, an R function to be called to invoke the browser.<br>Under Windows NULL is also allowed (and is the default), and implies that the file association mechanism will be used. |
| encodeIfNeeded | Should the URL be encoded by URLencode before passing to the browser? This is not needed (and might be harmful) if the browser program/function itself does encoding, and can be harmful for 'file://' URLs on some systems and for 'http://' URLs passed to some CGI applications. Fortunately, most URLs do not need encoding. |

## Value

A browser is opened with the B.C. Data Catalogue URL loaded if the session is interactive. The URL used is returned as a character string.

## See Also

browseURL

## Examples

```
## Take me to the B.C. Data Catalogue home page
try(
  bcdc_browse()
)

## Take me to the B.C. airports catalogue record
try(
 bcdc_browse("bc-airports")
)

## Take me to the B.C. airports catalogue record
try(
  bcdc_browse("76b1b7a3-2112-4444-857a-afccf7b20da8")
)
```

`bcdc_check_geom_size` *Check spatial objects for WFS spatial operations*

## Description

Check a spatial object to see if it exceeds the current set value of 'bcdata.max_geom_pred_size' option, which controls how the object is treated when used inside a spatial predicate function in `filter.bcdc_promise()`. If the object does exceed the size threshold a bounding box is drawn around it and all features within the box will be returned. Further options include:

- Try adjusting the value of the 'bcdata.max_geom_pred_size' option
- Simplify the spatial object to reduce its size
- Further processing on the returned object

## Usage

```
bcdc_check_geom_size(x)
```

## Arguments

| | |
|---|---|
| `x` | object of class sf, sfc or sfg |

## Details

See the Querying Spatial Data with bcdata for more details.

## Value

invisibly return logical indicating whether the check pass. If the return value is TRUE, the object will not need a bounding box drawn. If the return value is FALSE, the check will fails and a bounding box will be drawn.

## Examples

```
try({
  airports <- bcdc_query_geodata("bc-airports") %>% collect()
  bcdc_check_geom_size(airports)
})
```

bcdc_describe_feature *Describe the attributes of a Web Feature Service*

## Description

Describe the attributes of column of a record accessed through the Web Feature Service. This can be a useful tool to examine a layer before issuing a query with bcdc_query_geodata.

## Usage

```
bcdc_describe_feature(record)
```

## Arguments

| | |
|---|---|
| record | either a bcdc_record object (from the result of bcdc_get_record()), a character string denoting the name or ID of a resource (or the URL) or a BC Geographic Warehouse (BCGW) name.<br>It is advised to use the permanent ID for a record or the BCGW name rather than the human-readable name to guard against future name changes of the record. If you use the human-readable name a warning will be issued once per session. You can silence these warnings altogether by setting an option: options("silence_named_get_data_warning" = TRUE) - which you can set in your .Rprofile file so the option persists across sessions. |

## Value

bcdc_describe_feature returns a tibble describing the attributes of a B.C. Data Catalogue record. The tibble returns the following columns:

- col_name: attributes of the feature
- sticky: whether a column can be separated from the record in a Web Feature Service call via the dplyr::select method
- remote_col_type: class of what is return by the web feature service
- local_col_type: the column class in R
- column_comments: additional metadata specific to that column

## Examples

```
try(
  bcdc_describe_feature("bc-airports")
)

try(
  bcdc_describe_feature("WHSE_IMAGERY_AND_BASE_MAPS.GSR_AIRPORTS_SVW")
)
```

`bcdc_get_citation` *Generate a bibentry from a Data Catalogue Record*

## Description

Generate a "TechReport" bibentry object directly from a catalogue record. The primary use of this function is as a helper to create a `.bib` file for use in reference management software to cite data from the B.C. Data Catalogue. This function is likely to be starting place for this process and manual adjustment will often be needed. The bibentries are not designed to be authoritative and may not reflect all fields required for individual citation requirements.

## Usage

```
bcdc_get_citation(record)
```

## Arguments

| | |
|---|---|
| `record` | either a `bcdc_record` object (from the result of `bcdc_get_record()`), a character string denoting the name or ID of a resource (or the URL)<br>It is advised to use the permanent ID for a record rather than the human-readable name to guard against future name changes of the record. If you use the human-readable name a warning will be issued once per session. You can silence these warnings altogether by setting an option: `options("silence_named_get_data_warning" = TRUE)` - which you can set in your .Rprofile file so the option persists across sessions. |

## See Also

`utils::bibentry()`

## Examples

```
try(
 bcdc_get_citation("76b1b7a3-2112-4444-857a-afccf7b20da8")
)

## Or directly on a record object
try(
 bcdc_get_citation(bcdc_get_record("76b1b7a3-2112-4444-857a-afccf7b20da8"))
)
```

bcdc_get_data *Download and read a resource from a B.C. Data Catalogue record*

## Description

Download and read a resource from a B.C. Data Catalogue record

## Usage

```
bcdc_get_data(record, resource = NULL, verbose = TRUE, ...)
```

## Arguments

| | |
|---|---|
| record | either a bcdc_record object (from the result of bcdc_get_record()), a character string denoting the name or ID of a resource (or the URL) or a BC Geographic Warehouse (BCGW) name.<br>It is advised to use the permanent ID for a record or the BCGW name rather than the human-readable name to guard against future name changes of the record. If you use the human-readable name a warning will be issued once per session. You can silence these warnings altogether by setting an option: options("silence_named_get_data_warning" = TRUE) - which you can set in your .Rprofile file so the option persists across sessions. |
| resource | optional argument used when there are multiple data files within the same record. See examples. |
| verbose | When more than one resource is available for a record, should extra information about those resources be printed to the console? Default TRUE |
| ... | arguments passed to other functions. Tabular data is passed to a function to handle the import based on the file extension. bcdc_read_functions() provides details on which functions handle the data import. You can then use this information to look at the help pages of those functions. See the examples for a workflow that illustrates this process. For spatial Web Feature Service data the ... arguments are passed to bcdc_query_geodata(). |

## Value

An object of a type relevant to the resource (usually a tibble or an sf object, a list if the resource is a json file)

## Examples

```
# Using the record and resource ID:
try(
  bcdc_get_data(record = '76b1b7a3-2112-4444-857a-afccf7b20da8',
                resource = '4d0377d9-e8a1-429b-824f-0ce8f363512c')
)

try(
```

```
  bcdc_get_data('2a5d2e5e-36eb-412b-89ff-508dae26a7ff')
)

# Using a `bcdc_record` object obtained from `bcdc_get_record`:
try(
  record <- bcdc_get_record('2a5d2e5e-36eb-412b-89ff-508dae26a7ff')
)

try(
  bcdc_get_data(record)
)

# Using a BCGW name
try(
  bcdc_get_data("WHSE_IMAGERY_AND_BASE_MAPS.GSR_AIRPORTS_SVW")
)

# Using sf's sql querying ability
try(
  bcdc_get_data(
    record = '30aeb5c1-4285-46c8-b60b-15b1a6f4258b',
    resource = '3d72cf36-ab53-4a2a-9988-a883d7488384',
    layer = 'BC_Boundary_Terrestrial_Line',
   query = "SELECT SHAPE_Length, geom FROM BC_Boundary_Terrestrial_Line WHERE SHAPE_Length < 100"
  )
)

## Example of correcting import problems

## Some initial problems reading in the data
try(
  bcdc_get_data('d7e6c8c7-052f-4f06-b178-74c02c243ea4')
)

## From bcdc_get_record we realize that the data is in xlsx format
try(
 bcdc_get_record('8620ce82-4943-43c4-9932-40730a0255d6')
)

## bcdc_read_functions let's us know that bcdata
## uses readxl::read_excel to import xlsx files
try(
 bcdc_read_functions()
)

## bcdata let's you know that this resource has
## multiple worksheets
try(
 bcdc_get_data('8620ce82-4943-43c4-9932-40730a0255d6')
)

## we can control what is read in from an excel file
## using arguments from readxl::read_excel
```

```
try(
  bcdc_get_data('8620ce82-4943-43c4-9932-40730a0255d6', sheet = 'Purpose Built Rental')
)


## Pass an argument through to a read_* function

try(
  bcdc_get_data(record = "a2a2130b-e853-49e8-9b30-1d0c735aa3d9",
                resource = "0b9e7d31-91ff-4146-a473-106a3b301964")
)

## we can control some properties of the list object returned by
## jsonlite::read_json by setting simplifyVector = TRUE or
## simplifyDataframe = TRUE
try(
 bcdc_get_data(record = "a2a2130b-e853-49e8-9b30-1d0c735aa3d9",
                resource = "0b9e7d31-91ff-4146-a473-106a3b301964",
                simplifyVector = TRUE)
)
```

---

`bcdc_get_record` *Show a single B.C. Data Catalogue record*

---

## Description

Show a single B.C. Data Catalogue record

## Usage

```
bcdc_get_record(id)
```

## Arguments

| | |
|---|---|
| `id` | the human-readable name, permalink ID, or URL of the record.<br>It is advised to use the permanent ID for a record rather than the human-readable name to guard against future name changes of the record. If you use the human-readable name a warning will be issued once per session. You can silence these warnings altogether by setting an option: `options("silence_named_get_record_warning" = TRUE)` - which you can put in your .Rprofile file so the option persists across sessions. |

## Value

A list containing the metadata for the record

## Examples

```
try(
  bcdc_get_record("https://catalogue.data.gov.bc.ca/dataset/bc-airports")
)

try(
  bcdc_get_record("bc-airports")
)

try(
 bcdc_get_record("https://catalogue.data.gov.bc.ca/dataset/76b1b7a3-2112-4444-857a-afccf7b20da8")
)

try(
  bcdc_get_record("76b1b7a3-2112-4444-857a-afccf7b20da8")
)
```

---

`bcdc_list` *Return a full list of the names of B.C. Data Catalogue records*

---

## Description

Return a full list of the names of B.C. Data Catalogue records

## Usage

```
bcdc_list()
```

## Value

A character vector of the names of B.C. Data Catalogue records

## Examples

```
try(
  bcdc_list()
)
```

## bcdc_list_groups *Retrieve group information for B.C. Data Catalogue*

### Description

Returns a tibble of groups or records. Groups can be viewed here: https://catalogue.data.gov.bc.ca/group or accessed directly from R using `bcdc_list_groups`

### Usage

```
bcdc_list_groups()

bcdc_list_group_records(group)
```

### Arguments

| | |
|---|---|
| `group` | Name of the group |

### Functions

- `bcdc_list_groups()`: List the available groups in the B.C. Data Catalogue.

### Examples

```
try(
  bcdc_list_group_records('environmental-reporting-bc')
)
```

## bcdc_list_organizations *Retrieve organization information for B.C. Data Catalogue*

### Description

Returns a tibble of organizations or records. Organizations can be viewed here: https://catalogue.data.gov.bc.ca/organizations or accessed directly from R using `bcdc_list_organizations`

### Usage

```
bcdc_list_organizations()

bcdc_list_organization_records(organization)
```

### Arguments

| | |
|---|---|
| `organization` | Name of the organization |

## Functions

- `bcdc_list_organizations()`: List the available organizations in the B.C. Data Catalogue.

## Examples

```
try(
  bcdc_list_organization_records('bc-stats')
)
```

---

`bcdc_options` *Retrieve options used in bcdata, their value if set and the default value.*

---

## Description

This function retrieves bcdata specific options that can be set. These options can be set using `option({name of the option} = {value of the option})`. The default options are purposefully set conservatively to hopefully ensure successful requests. Resetting these options may result in failed calls to the data catalogue. Options in R are reset every time R is re-started. See examples for additional ways to restore your initial state.

## Usage

```
bcdc_options()
```

## Details

`bcdata.max_geom_pred_size` is the maximum size in bytes of an object used for a geometric operation. Objects that are bigger than this value will have a bounding box drawn and apply the geometric operation on that simpler polygon. The bcdc_check_geom_size function can be used to assess whether a given spatial object exceeds the value of this option. Users can iteratively try to increase the maximum geometric predicate size and see if the bcdata catalogue accepts the request.

`bcdata.chunk_limit` is an option useful when dealing with very large data sets. When requesting large objects from the catalogue, the request is broken up into smaller chunks which are then re-combined after they've been downloaded. This is called "pagination". bcdata does this all for you, however by using this option you can set the size of the chunk requested. On slower connections, or when having problems, it may help to lower the chunk limit.

`bcdata.max_package_search_limit` is an option for setting the maximum number of datasets returned when querying by organization with the package_search API endpoint. The default limit (1000) is purposely set high to return all datasets for a given organization.

`bcdata.max_package_search_facet_limit` is an option for setting the maximum number of values returned when querying facet fields with the package_search API endpoint. The default limit (1000) is purposely set high to return all values for each facet field ("license_id", "download_audience", "res_format", "publish_state", "organization", "groups").

`bcdata.max_group_package_show_limit` is an option for setting the maximum number of datasets returned when querying by group with the group_package_show API endpoint. The default limit (1000) is purposely set high to return all datasets for a given group.

`bcdata.single_download_limit` *Deprecated*. This is the maximum number of records an object can be before forcing a paginated download; it is set by querying the server capabilities. This option is deprecated and will be removed in a future release. Use `bcdata.chunk_limit` to set a lower value pagination value.

## Examples

```
## Save initial conditions
try(
  original_options <- options()
)

## See initial options
try(
  bcdc_options()
)

try(
  options(bcdata.max_geom_pred_size = 1E6)
)

## See updated options
try(
  bcdc_options()
)

## Reset initial conditions
try(
  options(original_options)
)
```

---

`bcdc_preview` *Get preview map from the B.C. Web Map Service*

---

## Description

Note this does not return the actual map features, rather opens an image preview of the layer in a Leaflet map window

## Usage

```
bcdc_preview(record)
```

## Arguments

| | |
|---|---|
| record | either a bcdc_record object (from the result of bcdc_get_record()), a character string denoting the name or ID of a resource (or the URL) or a BC Geographic Warehouse (BCGW) name.<br>It is advised to use the permanent ID for a record or the BCGW name rather than the human-readable name to guard against future name changes of the record. If you use the human-readable name a warning will be issued once per session. You can silence these warnings altogether by setting an option: options("silence_named_get_data_warning" = TRUE) - which you can set in your .Rprofile file so the option persists across sessions. |

## Examples

```
try(
  bcdc_preview("regional-districts-legally-defined-administrative-areas-of-bc")
)

try(
  bcdc_preview("water-reservations-points")
)

# Using BCGW name
try(
  bcdc_preview("WHSE_LEGAL_ADMIN_BOUNDARIES.ABMS_REGIONAL_DISTRICTS_SP")
)
```

---

# bcdc_query_geodata *Query data from the B.C. Web Feature Service*

---

## Description

Queries features from the B.C. Web Feature Service. See bcdc_tidy_resources() - if a resource has a value of "wms" in the format column it is available as a Web Feature Service, and you can query and download it using bcdc_query_geodata(). The response will be paginated if the number of features is greater than that allowed by the server. Please see bcdc_options() for defaults and more information.

## Usage

```
bcdc_query_geodata(record, crs = 3005)
```

## Arguments

| | |
|---|---|
| record | either a bcdc_record object (from the result of bcdc_get_record()), a character string denoting the name or ID of a resource (or the URL) or a BC Geographic Warehouse (BCGW) name. |

It is advised to use the permanent ID for a record or the BCGW name rather than the human-readable name to guard against future name changes of the record. If you use the human-readable name a warning will be issued once per session. You can silence these warnings altogether by setting an option: `options("silence_named_get_data_warning" = TRUE)` - which you can set in your .Rprofile file so the option persists across sessions.

`crs` the epsg code for the coordinate reference system. Defaults to `3005` (B.C. Albers). See https://epsg.io.

## Details

Note that this function doesn't actually return the data, but rather an object of class `bcdc_promise`, which includes all of the information required to retrieve the requested data. In order to get the actual data as an `sf` object, you need to run `collect()` on the `bcdc_promise`. This allows further refining the call to `bcdc_query_geodata()` with `filter()` and/or `select()` statements before pulling down the actual data as an `sf` object with `collect()`. See examples.

## Value

A `bcdc_promise` object. This object includes all of the information required to retrieve the requested data. In order to get the actual data as an `sf` object, you need to run `collect()` on the `bcdc_promise`.

## Examples

```
# Returns a bcdc_promise, which can be further refined using filter/select:
try(
  res <- bcdc_query_geodata("bc-airports", crs = 3857)
)

# To obtain the actual data as an sf object, collect() must be called:
try(
  res <- bcdc_query_geodata("bc-airports", crs = 3857) %>%
    filter(PHYSICAL_ADDRESS == 'Victoria, BC') %>%
    collect()
)

# To query based on partial matches, use %LIKE%:
try(
  res <- bcdc_query_geodata("bc-airports") %>%
    filter(PHYSICAL_ADDRESS %LIKE% 'Vict%')
)

# To query using %IN%
try(
  res <- bcdc_query_geodata("bc-airports") %>%
    filter(
      AIRPORT_NAME %IN%
        c(
          "Victoria Harbour (Camel Point) Heliport",
```

```
          "Victoria Harbour (Shoal Point) Heliport"
        )
    ) %>%
    collect()
)


try(
  res <- bcdc_query_geodata("groundwater-wells") %>%
    filter(OBSERVATION_WELL_NUMBER == "108") %>%
    select(WELL_TAG_NUMBER, INTENDED_WATER_USE) %>%
    collect()
)

## A moderately large layer
try(
  res <- bcdc_query_geodata("bc-environmental-monitoring-locations")
)

try(
  res <- bcdc_query_geodata("bc-environmental-monitoring-locations") %>%
    filter(PERMIT_RELATIONSHIP == "DISCHARGE")
)


## A very large layer
try(
 res <- bcdc_query_geodata("terrestrial-protected-areas-representation-by-biogeoclimatic-unit")
)

## Using a BCGW name
try(
  res <- bcdc_query_geodata("WHSE_IMAGERY_AND_BASE_MAPS.GSR_AIRPORTS_SVW")
)
```

---

`bcdc_read_functions` *Formats supported and loading functions*

---

## Description

Provides a tibble of formats supported by bcdata and the associated function that reads that data into R. This function is meant as a resource to determine which parameters can be passed through the `bcdc_get_data` function to the reading function. This is particularly important to know if the data requires using arguments from the read in function.

## Usage

```
bcdc_read_functions()
```

bcdc_search *Search the B.C. Data Catalogue*

## Description

Search the B.C. Data Catalogue

## Usage

```
bcdc_search(
  ...,
  license_id = NULL,
  download_audience = NULL,
  res_format = NULL,
  sector = NULL,
  organization = NULL,
  groups = NULL,
  n = 100
)
```

## Arguments

| | |
|---|---|
| `...` | search terms |
| `license_id` | the type of license (see `bcdc_search_facets("license_id")`). |
| `download_audience` | download audience (see `bcdc_search_facets("download_audience")`). Default `NULL` (all audiences). |
| `res_format` | format of resource (see `bcdc_search_facets("res_format")`) |
| `sector` | sector of government from which the data comes (see `bcdc_search_facets("sector")`) |
| `organization` | government organization that manages the data (see `bcdc_search_facets("organization")`) |
| `groups` | collections of datasets for a particular project or on a particular theme (see `bcdc_search_facets("groups")`) |
| `n` | number of results to return. Default `100` |

## Value

A list containing the records that match the search

## Examples

```
try(
  bcdc_search("forest")
)

try(
  bcdc_search("regional district", res_format = "fgdb")
```

```
)

try(
  bcdc_search("angling", groups = "bc-tourism")
)
```

---

bcdc_search_facets *Get the valid values for a facet (that you can use in* bcdc_search() *)*

---

## Description

Get the valid values for a facet (that you can use in bcdc_search())

## Usage

```
bcdc_search_facets(
  facet = c("license_id", "download_audience", "res_format", "publish_state",
    "organization", "groups")
)
```

## Arguments

| | |
|---|---|
| facet | the facet(s) for which to retrieve valid values. Can be one or more of: "license_id", "download_audier |

## Value

A data frame of values for the selected facet

## Examples

```
try(
  bcdc_search_facets("download_audience")
)

try(
  bcdc_search_facets("res_format")
)
```

`bcdc_tidy_resources` *Provide a data frame containing the metadata for all resources from a single B.C. Data Catalogue record*

## Description

Returns a rectangular data frame of all resources contained within a record. This is particularly useful if you are trying to construct a vector of multiple resources in a record. The data frame also provides useful information on the formats, availability and types of data available.

## Usage

```
bcdc_tidy_resources(record)
```

## Arguments

| | |
|---|---|
| `record` | either a `bcdc_record` object (from the result of `bcdc_get_record()`), a character string denoting the name or ID of a resource (or the URL) or a BC Geographic Warehouse (BCGW) name.<br>It is advised to use the permanent ID for a record or the BCGW name rather than the human-readable name to guard against future name changes of the record. If you use the human-readable name a warning will be issued once per session. You can silence these warnings altogether by setting an option: `options("silence_named_get_data_warning" = TRUE)` - which you can set in your .Rprofile file so the option persists across sessions. |

## Value

A data frame containing the metadata for all the resources for a record

## Examples

```
try(
  airports <- bcdc_get_record("bc-airports")
)

try(
  bcdc_tidy_resources(airports)
)
```

## CQL *CQL escaping*

### Description

Write a CQL expression to escape its inputs, and return a CQL/SQL object. Used when writing filter expressions in `bcdc_query_geodata()`.

### Usage

```
CQL(...)
```

### Arguments

| | |
|---|---|
| ... | Character vectors that will be combined into a single CQL statement. |

### Details

See the CQL/ECQL for Geoserver website.

### Value

An object of class `c("CQL", "SQL")`

### Examples

```
CQL("FOO > 12 & NAME LIKE 'A&'")
```

## cql_geom_predicates *CQL Geometry Predicates*

### Description

Functions to construct a CQL expression to be used to filter results from `bcdc_query_geodata()`. See the geoserver CQL documentation for details. The sf object is automatically converted in a bounding box to reduce the complexity of the Web Feature Service call. Subsequent in-memory filtering may be needed to achieve exact results.

## Usage

```
EQUALS(geom)

DISJOINT(geom)

INTERSECTS(geom)

TOUCHES(geom)

CROSSES(geom)

WITHIN(geom)

CONTAINS(geom)

OVERLAPS(geom)

BBOX(coords, crs = NULL)

DWITHIN(
  geom,
  distance,
  units = c("meters", "feet", "statute miles", "nautical miles", "kilometers")
)
```

## Arguments

| | |
|---|---|
| geom | an sf/sfc/sfg or bbox object (from the sf package) |
| coords | the coordinates of the bounding box as four-element numeric vector c(xmin, ymin, xmax, ymax), a bbox object from the sf package (the result of running sf::st_bbox() on an sf object), or an sf object which then gets converted to a bounding box on the fly. |
| crs | (Optional) A numeric value or string containing an SRS code. If coords is a bbox object with non-empty crs, it is taken from that. (For example, 'EPSG:3005' or just 3005. The default is to use the CRS of the queried layer) |
| distance | numeric value for distance tolerance |
| units | units that distance is specified in. One of "feet", "meters", "statute miles", "nautical miles", "kilometers" |

## Value

a CQL expression to be passed on to the WFS call

# Index


BBOX *(cql_geom_predicates)*, 20
bcdc_browse, 2
bcdc_check_geom_size, 4, *12*
bcdc_describe_feature, 5
bcdc_get_citation, 6
bcdc_get_data, 7
bcdc_get_record, 9
bcdc_list, 10
bcdc_list_group_records *(bcdc_list_groups)*, 11
bcdc_list_groups, 11
bcdc_list_organization_records *(bcdc_list_organizations)*, 11
bcdc_list_organizations, 11
bcdc_options, 12
bcdc_options(), *14*
bcdc_preview, 13
bcdc_query_geodata, 14
bcdc_query_geodata(), *7*, *20*
bcdc_read_functions, 16
bcdc_read_functions(), *7*
bcdc_search, 17
bcdc_search(), *18*
bcdc_search_facets, 18
bcdc_tidy_resources, 19
bcdc_tidy_resources(), *14*
browseURL, *3*

collect(), *15*
CONTAINS *(cql_geom_predicates)*, 20
CQL, 20
CQL-class *(CQL)*, 20
cql_geom_predicates, 20
CROSSES *(cql_geom_predicates)*, 20

DISJOINT *(cql_geom_predicates)*, 20
DWITHIN *(cql_geom_predicates)*, 20

EQUALS *(cql_geom_predicates)*, 20

filter(), *15*
filter.bcdc_promise(), *4*

INTERSECTS *(cql_geom_predicates)*, 20

OVERLAPS *(cql_geom_predicates)*, 20

select(), *15*

TOUCHES *(cql_geom_predicates)*, 20

URLencode, *3*
utils::bibentry(), *6*

WITHIN *(cql_geom_predicates)*, 20