

Package: `edr4r` (via `r-universe`)

June 18, 2026

Title Client for the 'OGC API - Environmental Data Retrieval' Standard

Version 0.1.0

Description A tidy 'R' client for services implementing the 'OGC API - Environmental Data Retrieval' ('EDR') standard. It is general purpose, but most of its real-world use is against in-situ monitoring networks (stream gauges, weather stations, snow and reservoir telemetry) that expose their stations and time series as 'EDR' collections. Known working endpoints include the 'USGS waterdata OGC API' and the 'Western Water Datahub'. Provides discovery, query, and parsing helpers for the locations, items, position, area, cube, radius, trajectory, and corridor query types. Returns 'CoverageJSON' as tidy 'tibble' rows and 'GeoJSON' as 'sf' objects.

License MIT + file LICENSE

URL <https://github.com/ksonda/edr4r>

BugReports <https://github.com/ksonda/edr4r/issues>

Encoding UTF-8

Depends R (>= 4.1)

Imports cli, httr2 (>= 1.0.0), jsonlite, purrr, rlang, tibble, vctrs

Suggests base64enc, ggplot2, htmlwidgets, knitr, leaflet, rmarkdown, sf, svglite, testthat (>= 3.0.0)

Config/testthat/edition 3

RoxygenNote 7.3.3

VignetteBuilder knitr

NeedsCompilation no

Author Kyle Onda [aut, cre, cph]

Maintainer Kyle Onda <konda@lincolninst.edu>

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

Date/Publication 2026-06-18 17:35:55 UTC

RemoteUrl <https://github.com/cran/edr4r>

RemoteRef HEAD

RemoteSha f2f9d6c5df8fefb92a909b49d98e5c750248459b

Contents

covjson_to_tibble	2
edr_area	3
edr_client	4
edr_collection	5
edr_collections	5
edr_conformance	6
edr_corridor	6
edr_cube	7
edr_explore	8
edr_items	10
edr_landing	11
edr_location	12
edr_locations	13
edr_map	14
edr_parameter_groups	16
edr_parameters	17
edr_plot	17
edr_position	19
edr_queryables	20
edr_radius	20
edr_request	21
edr_save_html	22
edr_trajectory	23
geojson_to_sf	24

Index **25**

covjson_to_tibble	<i>Convert a CoverageJSON response to a tidy tibble</i>
-------------------	---

Description

Flattens a CoverageJSON Coverage or CoverageCollection into a long tibble with one row per range value and axis-coordinate combination. Handles the Point and PointSeries domain types used by station-based EDR providers, and falls back to a general N-dimensional unrolling for Grid-like domains.

Usage

```
covjson_to_tibble(x, datetime_as_posix = TRUE)
```

Arguments

x	A CoverageJSON object: either an <code>edr_response</code> returned by <code>edr_location()</code> / <code>edr_area()</code> / <code>edr_cube()</code> (etc.) with <code>format = "covjson"</code> , or the raw parsed list.
<code>datetime_as_posix</code>	If TRUE (default), attempts to parse the time axis to POSIXct (UTC), including RFC 3339 UTC offsets. If any non-missing value cannot be parsed, warns and preserves the whole axis as character.

Value

A tibble with columns `coverage_id`, `parameter`, `parameter_label`, `unit`, `datetime`, `x`, `y`, `z`, and `value`. Columns that are absent from the source are filled with NA.

<code>edr_area</code>	<i>Area query (data inside a polygon)</i>
-----------------------	---

Description

Calls GET `/collections/{collection_id}/area` with a WKT POLYGON in the `coords` parameter.

Usage

```
edr_area(
  client,
  collection_id,
  coords,
  datetime = NULL,
  parameter_name = NULL,
  z = NULL,
  crs = NULL,
  format = c("covjson", "json"),
  ...
)
```

Arguments

<code>client</code>	An <code>edr_client</code> .
<code>collection_id</code>	Collection identifier.
<code>coords</code>	WKT polygon string, or a matrix / data.frame of (lon, lat) rows that will be closed into a POLYGON. May also be an <code>sf</code> / <code>sfc</code> polygon if <code>sf</code> is installed.
<code>datetime</code>	ISO-8601 instant or interval, e.g. "2024-01-01/2024-12-31" or "2024-01-01/..".
<code>parameter_name</code>	Character vector of parameter names to filter on. Sent as a comma-separated <code>parameter-name= query</code> .

z	Vertical level filter.
crs	Optional CRS URI for the response.
format	"covjson" (default), "geojson", "csv", or "json".
...	Additional query parameters passed through verbatim.

Value

An `edr_response` wrapping the returned CoverageJSON, or a parsed list for other JSON. The result contains data inside the requested polygon.

edr_client	<i>Create an EDR client</i>
------------	-----------------------------

Description

Builds a reusable client object that captures the base URL of an OGC API - EDR service, a default user-agent, and HTTP options that are applied to every request.

Usage

```
edr_client(
    base_url,
    user_agent = NULL,
    timeout = 60,
    max_tries = 3,
    headers = NULL,
    verbose = FALSE
)
```

Arguments

base_url	Base URL of an OGC API - EDR service. Examples: the USGS waterdata OGC API at "https://api.waterdata.usgs.gov/ogcapi/beta", the Western Water Datahub at "https://api.wwdh.internetofwater.app", or "http://localhost:5005" for a local pygeoapi dev server. A trailing slash is optional.
user_agent	String sent in the User-Agent header. Defaults to "edr4r/<version> (+https://github.com/ksonda/
timeout	Positive request timeout in seconds. Defaults to 60.
max_tries	Maximum number of attempts per request. The client retries on 408, 429, and 5xx responses with exponential backoff. Must be a positive integer; defaults to 3.
headers	Named character vector of extra headers attached to every request (e.g. c(Authorization = "Bearer ...").
verbose	If TRUE, prints request URLs to the console as they are made. Useful for debugging.

Value

An object of class `edr_client` containing the service URL and reusable HTTP request settings.

Examples

```
usgs <- edr_client("https://api.waterdata.usgs.gov/ogcapi/beta")
usgs
```

<code>edr_collection</code>	<i>Get a single collection's metadata</i>
-----------------------------	---

Description

Get a single collection's metadata

Usage

```
edr_collection(client, collection_id)
```

Arguments

<code>client</code>	An <code>edr_client</code> .
<code>collection_id</code>	Collection identifier as advertised by the server – e.g. "monitoring-locations" or "daily-values".

Value

A list with the raw collection document.

<code>edr_collections</code>	<i>List collections offered by the service</i>
------------------------------	--

Description

List collections offered by the service

Usage

```
edr_collections(client)
```

Arguments

<code>client</code>	An <code>edr_client</code> .
---------------------	------------------------------

Value

A tibble with one row per collection. Always includes `id`, `title`, `description`, `extent_bbox`, `crs`, `data_queries`, and `links` columns.

edr_conformance	<i>Declared OGC API conformance classes</i>
-----------------	---

Description

Declared OGC API conformance classes

Usage

```
edr_conformance(client)
```

Arguments

client	An edr_client.
--------	----------------

Value

A character vector of conformance class URIs.

edr_corridor	<i>Corridor query (data along a path with a width)</i>
--------------	--

Description

Calls GET /collections/{collection_id}/corridor.

Usage

```
edr_corridor(
  client,
  collection_id,
  coords,
  corridor_width,
  corridor_height = NULL,
  width_units = "km",
  height_units = "m",
  datetime = NULL,
  parameter_name = NULL,
  z = NULL,
  crs = NULL,
  format = c("covjson", "json"),
  ...
)
```

Arguments

client	An edr_client.
collection_id	Collection identifier.
coords	WKT LINESTRING, a matrix / data.frame of (lon, lat) rows, or an sfc linestring.
corridor_width	Positive width of the corridor.
corridor_height	Optional positive vertical extent.
width_units	Units for corridor_width.
height_units	Units for corridor_height.
datetime	ISO-8601 instant or interval, e.g. "2024-01-01/2024-12-31" or "2024-01-01/..".
parameter_name	Character vector of parameter names to filter on. Sent as a comma-separated parameter-name= query.
z	Vertical level filter.
crs	Optional CRS URI for the response.
format	"covjson" (default), "geojson", "csv", or "json".
...	Additional query parameters passed through verbatim.

Value

An edr_response wrapping the returned CoverageJSON, or a parsed list for other JSON. The result contains data inside the requested corridor around the path.

edr_cube	<i>Cube query (data inside a bounding box)</i>
----------	--

Description

Calls GET /collections/{collection_id}/cube with a bounding box.

Usage

```
edr_cube(
  client,
  collection_id,
  bbox,
  datetime = NULL,
  parameter_name = NULL,
  z = NULL,
  crs = NULL,
  format = c("covjson", "json"),
  ...
)
```

Arguments

client	An edr_client.
collection_id	Collection identifier.
bbox	Numeric vector of length 4 or 6.
datetime	ISO-8601 instant or interval, e.g. "2024-01-01/2024-12-31" or "2024-01-01/..".
parameter_name	Character vector of parameter names to filter on. Sent as a comma-separated parameter-name= query.
z	Vertical level filter.
crs	Optional CRS URI for the response.
format	"covjson" (default), "geojson", "csv", or "json".
...	Additional query parameters passed through verbatim.

Value

An edr_response wrapping the returned CoverageJSON, or a parsed list for other JSON. The result contains data inside the requested bounding box.

edr_explore	<i>One-shot fetch + plot + map for a collection</i>
-------------	---

Description

Convenience wrapper that finds stations via `edr_locations()`, fetches time series with **one** bulk request via `edr_cube()` or `edr_area()` when the collection supports it, and hands the lot to `edr_map()` for rendering. Optionally writes the map to a selfcontained HTML file.

Usage

```
edr_explore(
  client,
  collection_id,
  bbox = NULL,
  coords = NULL,
  datetime = NULL,
  parameter_name = NULL,
  limit = NULL,
  record_limit = NULL,
  file = NULL,
  popup = "plot+csv",
  method = c("auto", "cube", "area", "position", "per-location"),
  output = c("auto", "map", "plot", "data"),
  plot_view = c("auto", "time", "profile", "grid"),
  quiet = FALSE,
  ...
)
```

Arguments

client	An <code>edr_client</code> .
collection_id	Collection identifier.
bbox	Optional numeric length-4 bbox. Used both to filter the locations index (if the server honours it) and as the bbox for the cube fetch in <code>method = "auto"</code> . If omitted with <code>method = "cube"</code> , derived from the bounding box of the returned locations sf.
coords	Point coords for position, or polygon coords for area. Forwarded to <code>edr_position()</code> / <code>edr_area()</code> .
datetime	ISO-8601 interval forwarded to the data fetch.
parameter_name	Character vector of parameter ids; forwarded to the data fetch. Use <code>edr_parameters()</code> to discover valid ids.
limit	Optional cap on the number of stations to map.
record_limit	Optional per-station record cap, passed through to <code>edr_location()</code> in the per-location path. Useful for servers (e.g. USGS waterdata) that cap responses at ~10 records by default. Ignored on the cube and area paths.
file	If non-NULL, write the map to this HTML path via <code>edr_save_html()</code> and return file invisibly. Otherwise return the leaflet map.
popup	Popup mode (forwarded to <code>edr_map()</code>).
method	One of "auto" (default), "cube", "area", "position", or "per-location". See above.
output	One of "auto" (default), "map", "plot", or "data". "auto" returns a station map for station time-series results and an interactive coverage map for gridded/profile results. Explicit bulk methods returning "data" skip location discovery when the locations would not be used.
plot_view	Plot view passed to <code>edr_plot()</code> when returning a plot. Defaults to "auto".
quiet	If FALSE (default), print a cli progress bar when falling back to per-location fetches.
...	Forwarded to <code>edr_map()</code> when returning a map.

Details

The default `method = "auto"` picks the cheapest route the collection advertises in its `data_queries`:

- **cube** – one HTTP call returning a `CoverageCollection` across the whole bbox. Fast. Used when the collection supports cube *and* a bbox is supplied.
- **area** – like cube but uses a polygon. Used when `coords` is supplied and the collection supports area.
- **position** – one HTTP call at a point. Useful for vertical profiles returned by position queries.
- **per-location** – the fallback: one HTTP call per station via `edr_location()`. Slower (N+1), used when neither spatial bulk query is supported or the matching spatial input was not supplied. For polygon coords, locations are first restricted to the polygon.

Force a specific path by setting `method`. `coords` is required for area and position; if `method = "cube"` and `bbox` is omitted, the `bbox` is derived from the returned locations.

When polygon `coords` are supplied but the collection only advertises `locations`, `method = "auto"` falls back to `bbox`-filtered location discovery, filters those locations to the polygon, and fetches each matching location. If the collection advertises `area`, the area query is attempted first; an unavailable endpoint (HTTP 404, 405, or 501) triggers the same locations fallback. Explicit `method = "area"` instead requires the collection's `/area` endpoint.

Value

A leaflet `htmlwidget`, a `ggplot`, a tidy tibble/list when `output = "data"`, or `invisible(file)` when a map is saved.

Examples

```
## Not run:
cl <- edr_client("https://api.wwdh.internetofwater.app")

# One /cube call across a bbox -- fast.
edr_explore(
  cl, "rise-edr",
  bbox      = c(-116, 35.5, -114, 36.5),
  datetime  = "2023-01-01/2023-03-31",
  parameter_name = "3",
  file      = tempfile(fileext = ".html")
)

## End(Not run)
```

edr_items

Items (OGC API Features) helpers

Description

Many EDR servers expose an OGC API Features `/items` endpoint alongside the EDR queries. Behaviour varies: some deployments implement `items` as a full Features endpoint, others as a thin stub used only to register the collection as both EDR and Features. In the stub case, non-trivial data is usually obtained via the EDR queries (`edr_locations()`, `edr_area()`, `edr_cube()`, etc.).

Usage

```
edr_items(
  client,
  collection_id,
  bbox = NULL,
  datetime = NULL,
  limit = NULL,
  format = c("geojson", "json"),
```

```

    ...
  )

  edr_item(client, collection_id, item_id, format = c("geojson", "json"), ...)

```

Arguments

client	An edr_client.
collection_id	Collection identifier.
bbox	Finite numeric vector of length 4 or 6 (c(minx, miny, maxx, maxy) or with z). Lower bounds must not exceed upper bounds.
datetime	ISO-8601 instant or interval, e.g. "2024-01-01/2024-12-31" or "2024-01-01/..".
limit	Maximum number of features to return.
format	"geojson" (default) or "json".
...	Additional query parameters passed through verbatim.
item_id	Identifier of a single feature.

Value

When the server returns GeoJSON, an sf object if the sf package is installed, otherwise an edr_response wrapping the raw GeoJSON. For other JSON responses, a parsed list. The result describes the matching collection items from [edr_items\(\)](#) or the requested feature from [edr_item\(\)](#).

 edr_landing

EDR service landing page

Description

Retrieves the service root document, which advertises links to the collections, conformance, and openapi endpoints.

Usage

```
edr_landing(client)
```

Arguments

client	An edr_client.
--------	----------------

Value

A list with the parsed landing document.

edr_location	<i>Get data for a single location</i>
--------------	---------------------------------------

Description

Calls GET /collections/{collection_id}/locations/{location_id}. Typically returns CoverageJSON; pass the result to `covjson_to_tibble()` for a tidy data frame.

Usage

```
edr_location(
  client,
  collection_id,
  location_id,
  datetime = NULL,
  parameter_name = NULL,
  z = NULL,
  crs = NULL,
  format = c("covjson", "geojson", "csv", "json"),
  ...
)
```

Arguments

<code>client</code>	An <code>edr_client</code> .
<code>collection_id</code>	Collection identifier.
<code>location_id</code>	Identifier of the location, as advertised by the server. IDs vary by deployment: bare integers, alphanumeric station codes, or compound identifiers (e.g. colon-separated triplets used by some snow / forecast networks). Reserved characters are URL-encoded for you; a literal / is rejected because it cannot round-trip through HTTP path segments.
<code>datetime</code>	ISO-8601 instant or interval, e.g. "2024-01-01/2024-12-31" or "2024-01-01/..".
<code>parameter_name</code>	Character vector of parameter names to filter on. Sent as a comma-separated <code>parameter-name=query</code> .
<code>z</code>	Vertical level filter.
<code>crs</code>	Optional CRS URI for the response.
<code>format</code>	"covjson" (default), "geojson", "csv", or "json".
<code>...</code>	Additional query parameters passed through verbatim.

Value

An `edr_response` wrapping the returned CoverageJSON or GeoJSON, a tibble for CSV, or a parsed list for other JSON. The result contains data associated with the requested location.

edr_locations	<i>List locations in a collection</i>
---------------	---------------------------------------

Description

Calls GET /collections/{collection_id}/locations. With no extra filters, EDR servers typically return a GeoJSON FeatureCollection of available locations.

Usage

```
edr_locations(
  client,
  collection_id,
  bbox = NULL,
  datetime = NULL,
  parameter_name = NULL,
  crs = NULL,
  limit = NULL,
  format = c("geojson", "json"),
  ...
)
```

Arguments

client	An edr_client.
collection_id	Collection identifier.
bbox	Finite numeric vector of length 4 or 6 (c(minx, miny, maxx, maxy) or with z). Lower bounds must not exceed upper bounds.
datetime	ISO-8601 instant or interval, e.g. "2024-01-01/2024-12-31" or "2024-01-01/..".
parameter_name	Character vector of parameter names to filter on. Sent as a comma-separated parameter-name= query.
crs	Optional CRS URI for the response.
limit	Maximum number of features to return.
format	"geojson" (default) or "json".
...	Additional query parameters passed through verbatim.

Value

When the server returns GeoJSON, an sf object if the sf package is installed, otherwise an edr_response wrapping the raw GeoJSON. When the server returns CoverageJSON, an edr_response. The result describes the collection locations matching the supplied filters.

edr_map

*Map EDR locations or coverage data***Description**

Builds a [leaflet:leaflet](#) map of station features or gridded/profile CoverageJSON data. Station maps can show per-station popups with interactive time-series charts and CSV downloads. Coverage maps keep all supplied parameters, times, and vertical levels in the widget and expose in-map controls for choosing the active slice; grid cells open popups with a time-series chart for the clicked cell.

Usage

```
edr_map(
  locations,
  data = NULL,
  popup = c("plot+csv", "plot", "csv", "table", "all"),
  location_col = "coverage_id",
  id_col = NULL,
  label_col = NULL,
  parameter = NULL,
  plot_width = 7,
  plot_height = 3.5,
  plot_dpi = 72,
  tile_provider = "CartoDB.Positron",
  marker_radius = 6,
  matched_color = "#2C7FB8",
  unmatched_color = "#BBBBBB",
  show_unmatched = TRUE,
  legend = TRUE,
  max_match_distance = NULL,
  mode = c("auto", "stations", "grid", "profile"),
  controls = TRUE,
  initial = list(),
  grid_opacity = 0.75
)
```

Arguments

locations	An sf object from edr_locations() , an <code>edr_response</code> wrapping GeoJSON, or tidy coverage data from covjson_to_tibble() / a CoverageJSON <code>edr_response</code> .
data	See above. Defaults to NULL.
popup	One of "plot+csv" (default), "plot", "csv", "table", or "all".
location_col	Column in data carrying the location id when data is a single tibble. Default "coverage_id".

id_col	Column in locations to join on. If NULL, the function looks for "id" then "_id" then the first character column.
label_col	Column in locations used for the popup heading. If NULL, tries "name", "locationName", "title", then the detected id column.
parameter	Optional character vector restricting which parameters get plotted in each popup.
plot_width, plot_height	Popup chart dimensions in inches. Display size in pixels is plot_width * plot_dpi by plot_height * plot_dpi, with a larger minimum size for readable interactive popups.
plot_dpi	Display dots-per-inch for popup charts. Default 72; bump to 90+ if popups look small on hi-DPI displays.
tile_provider	Leaflet basemap. Default "CartoDB.Positron".
marker_radius	Marker radius in pixels for stations that have time-series data. Data-less stations are drawn one pixel smaller.
matched_color	Marker colour for stations that joined to a coverage in data. Default deep blue.
unmatched_color	Marker colour for stations without data (only relevant when data is supplied and show_unmatched = TRUE). Default light grey.
show_unmatched	If TRUE (default), data-less stations are drawn in unmatched_color so the user can see the full station network. Set to FALSE to drop them entirely. Ignored when data is NULL.
legend	If TRUE (default), add a legend distinguishing stations with data from those without. Suppressed automatically when there are no unmatched markers to label.
max_match_distance	Optional maximum coordinate distance for spatially matching data rows with x / y columns to stations. Units are those of the station coordinates. NULL (default) keeps the nearest-station fallback unlimited, but warns when matches are not exact.
mode	Map mode. "auto" (default) uses station markers for spatial feature inputs, grid cells for gridded coverage data, and profile markers for vertical profiles. Use "stations", "grid", or "profile" to force a mode.
controls	If TRUE (default), coverage maps include in-map controls for available slice dimensions (parameter, datetime, and z for grids).
initial	Named list of initial coverage-map selections, e.g. list(parameter = "temperature", datetime = "2024-01-01", z = 0).
grid_opacity	Fill opacity for gridded coverage cells.

Details

data can be one of:

- NULL – just markers with the sf attribute table as a popup (when popup = "table" or popup = "all").

- A long tibble (the output of `covjson_to_tibble()`) with one column matching the locations' id column. Set `location_col =` to the column in `data` that holds the location id and `id_col =` to the column in `locations`.
- A named list of tibbles, keyed by feature id. This is what `edr_explore()` passes when it fetches one time series per station — and the right shape when each station has its own CovJSON response, because server-assigned `coverage_ids` like "1" won't naturally match the feature id.

Value

A leaflet htmlwidget. Pass it to `edr_save_html()` to write a selfcontained HTML file.

`edr_parameter_groups` *List cross-collection parameter groups*

Description

Some EDR deployments advertise a top-level `parameterGroups` block on `/collections`. A group maps one environmental concept to the parameter identifiers used by multiple collections, which is useful when building comparable cross-provider queries.

Usage

```
edr_parameter_groups(client)
```

Arguments

`client` An `edr_client`.

Details

This is an optional deployment extension rather than a required EDR capability. Servers without `parameterGroups` return an empty tibble.

Value

A tibble with `name`, `description`, and `list-column` members.

edr_parameters	<i>List the data parameters a collection serves</i>
----------------	---

Description

Pulls the `parameter_names` block out of the collection document (GET `/collections/{id}`) and flattens it into a tidy tibble. These are the observed properties you can pass to `parameter_name =` on the query verbs (`edr_location()`, `edr_cube()`, etc.).

Usage

```
edr_parameters(client, collection_id)
```

Arguments

<code>client</code>	An <code>edr_client</code> .
<code>collection_id</code>	Collection identifier as advertised by the server – e.g. "monitoring-locations" or "daily-values".

Details

EDR servers vary in how they key the `parameter_names` dictionary (numeric IDs, short codes, etc.). The `id` column in the returned tibble is the value to pass back as `parameter_name`; the `name` column is the human-readable label.

Value

A tibble with one row per parameter. Columns: `id`, `name`, `description`, `unit_symbol`, `unit_label`, `observed_property`.

edr_plot	<i>Plot an EDR response as a ggplot</i>
----------	---

Description

Convenience wrapper around `ggplot2::ggplot()` for the long tibble returned by `covjson_to_tibble()`. Automatically chooses a sensible view for time series, vertical profiles, and x/y grids.

Usage

```
edr_plot(
  data,
  parameter = NULL,
  group = "coverage_id",
  facet = "parameter",
  scales = "free_y",
  geom = c("line", "point", "both"),
  facet_labels = TRUE,
  view = c("auto", "time", "profile", "grid")
)
```

Arguments

data	Either a tidy tibble from <code>covjson_to_tibble()</code> or an <code>edr_response</code> / <code>edr_covjson</code> object (which we flatten with <code>covjson_to_tibble()</code> for you).
parameter	Optional character vector restricting to a subset of parameters.
group	Column in data used for the colour aesthetic. Defaults to "coverage_id" (one colour per location). Set to NULL to disable.
facet	Column to facet by. Defaults to "parameter" so each variable gets its own panel; pass NULL to plot everything on one axis.
scales	<code>facet_wrap()</code> scales argument. Default "free_y" gives each parameter its own y-axis range.
geom	One of "line", "point", or "both".
facet_labels	If TRUE (default), facet strip labels include the unit (e.g. "discharge (ft ³ /s)").
view	Plot view. "auto" (default) detects grids from varying x and y, profiles from varying z, and otherwise falls back to a time-series view. Set to "time", "profile", or "grid" to force a specific layout.

Value

A ggplot object visualizing the selected parameter and view.

Examples

```
## Not run:
cl <- edr_client("https://api.wwdh.internetofwater.app")
resp <- edr_location(cl, "rise-edr",
  location_id = 3514,
  datetime = "2023-01-01/2023-06-30",
  parameter_name = "3")

edr_plot(resp)

## End(Not run)
```

edr_position	<i>Position query (data at a point)</i>
--------------	---

Description

Calls GET /collections/{collection_id}/position with a WKT POINT in the coords parameter.

Usage

```
edr_position(
  client,
  collection_id,
  coords,
  datetime = NULL,
  parameter_name = NULL,
  z = NULL,
  crs = NULL,
  format = c("covjson", "json"),
  ...
)
```

Arguments

client	An edr_client.
collection_id	Collection identifier.
coords	Either a length-2 numeric vector c(lon, lat), a length-3 vector c(lon, lat, z), or a WKT POINT string.
datetime	ISO-8601 instant or interval, e.g. "2024-01-01/2024-12-31" or "2024-01-01/..".
parameter_name	Character vector of parameter names to filter on. Sent as a comma-separated parameter-name= query.
z	Vertical level filter.
crs	Optional CRS URI for the response.
format	"covjson" (default), "geojson", "csv", or "json".
...	Additional query parameters passed through verbatim.

Value

An edr_response wrapping the returned CoverageJSON, or a parsed list for other JSON. The result contains data at the requested point.

edr_queryables	<i>Get the queryables (filter properties) for a collection</i>
----------------	--

Description

Returns the OGC API queryables document for a collection – a JSON Schema describing the filter properties the server exposes (this is typically used by OGC API Features for CQL2 / property-based filtering). It is **not** the right place to look up the data parameters / observed properties an EDR collection serves; for that, use [edr_parameters\(\)](#).

Usage

```
edr_queryables(client, collection_id)
```

Arguments

client	An <code>edr_client</code> .
collection_id	Collection identifier as advertised by the server – e.g. "monitoring-locations" or "daily-values".

Value

A list with the parsed queryables document.

edr_radius	<i>Radius query (data within a radius of a point)</i>
------------	---

Description

Calls GET /collections/{collection_id}/radius.

Usage

```
edr_radius(
  client,
  collection_id,
  coords,
  within,
  within_units = "km",
  datetime = NULL,
  parameter_name = NULL,
  z = NULL,
  crs = NULL,
  format = c("covjson", "json"),
  ...
)
```

Arguments

client	An edr_client.
collection_id	Collection identifier.
coords	Either a length-2 numeric vector <code>c(lon, lat)</code> , a length-3 vector <code>c(lon, lat, z)</code> , or a WKT POINT string.
within	Positive radius value.
within_units	Units of within (e.g. "km", "mi").
datetime	ISO-8601 instant or interval, e.g. "2024-01-01/2024-12-31" or "2024-01-01/..".
parameter_name	Character vector of parameter names to filter on. Sent as a comma-separated <code>parameter-name= query</code> .
z	Vertical level filter.
crs	Optional CRS URI for the response.
format	"covjson" (default), "geojson", "csv", or "json".
...	Additional query parameters passed through verbatim.

Value

An `edr_response` wrapping the returned CoverageJSON, or a parsed list for other JSON. The result contains data within the requested radius of the point.

edr_request	<i>Perform a low-level EDR request</i>
-------------	--

Description

Generally you should not need to call this directly: the high-level verbs (`edr_locations()`, `edr_area()`, etc.) build the path and query string for you. Use `edr_request()` when you need to hit a bespoke path or a non-standard parameter.

Usage

```
edr_request(
  client,
  path,
  query = list(),
  format = c("json", "geojson", "covjson", "csv", "html", "raw"),
  parse = TRUE
)
```

Arguments

client	An <code>edr_client</code> from <code>edr_client()</code> .
path	Path under the base URL (with or without leading slash), e.g. "collections/monitoring-locations/1
query	Named list of query parameters. Values may be scalars or vectors; vectors are joined with ", ". NULL entries are dropped.
format	Response format: one of "json" (default), "geojson", "covjson", "csv", "html", or "raw". Passed as ?f= (except "covjson", which is sent as ?f=json with a CoverageJSON Accept hint, since EDR servers return CovJSON via JSON).
parse	If TRUE (default), parses JSON / GeoJSON / CovJSON bodies into R structures. If FALSE, returns the raw <code>httr2</code> response.

Value

The parsed response body: usually a list, tibble, or `edr_response` representing the requested EDR resource. Returns an `httr2_response` when `parse = FALSE` or `format = "raw"`.

<code>edr_save_html</code>	<i>Save a map to a standalone HTML file</i>
----------------------------	---

Description

Thin wrapper around `htmlwidgets::saveWidget()` for the leaflet map returned by `edr_map()` or `edr_explore()`. With `selfcontained = TRUE` (the default), popup chart data and CSV download links live inside the file – no sidecar directory.

Usage

```
edr_save_html(map, file, selfcontained = TRUE, ...)
```

Arguments

map	A leaflet or <code>htmlwidget</code> .
file	Path to write to.
selfcontained	If TRUE, embed all assets in the file.
...	Forwarded to <code>htmlwidgets::saveWidget()</code> .

Value

Invisibly returns file.

edr_trajectory *Trajectory query (data along a path)*

Description

Calls GET /collections/{collection_id}/trajectory.

Usage

```
edr_trajectory(
  client,
  collection_id,
  coords,
  datetime = NULL,
  parameter_name = NULL,
  z = NULL,
  crs = NULL,
  format = c("covjson", "json"),
  ...
)
```

Arguments

client	An <code>edr_client</code> .
collection_id	Collection identifier.
coords	WKT LINESTRING, a matrix / data.frame of (lon, lat) rows, or an sfc linestring.
datetime	ISO-8601 instant or interval, e.g. "2024-01-01/2024-12-31" or "2024-01-01/..".
parameter_name	Character vector of parameter names to filter on. Sent as a comma-separated <code>parameter-name= query</code> .
z	Vertical level filter.
crs	Optional CRS URI for the response.
format	"covjson" (default), "geojson", "csv", or "json".
...	Additional query parameters passed through verbatim.

Value

An `edr_response` wrapping the returned CoverageJSON, or a parsed list for other JSON. The result contains data sampled along the requested path.

`geojson_to_sf`*Convert a GeoJSON EDR response to an sf object*

Description

Convert a GeoJSON EDR response to an sf object

Usage

```
geojson_to_sf(x)
```

Arguments

`x` An `edr_response` wrapping GeoJSON (e.g. from `edr_locations()`) or a raw parsed GeoJSON list.

Value

An sf object containing the GeoJSON feature properties and geometries. Requires the `sf` package. If `sf` is not installed, returns a tibble of feature properties without geometry and warns.

Index

`covjson_to_tibble`, 2
`covjson_to_tibble()`, 12, 14, 16–18

`edr_area`, 3
`edr_area()`, 3, 8–10, 21
`edr_client`, 4
`edr_client()`, 22
`edr_collection`, 5
`edr_collections`, 5
`edr_conformance`, 6
`edr_corridor`, 6
`edr_cube`, 7
`edr_cube()`, 3, 8, 10, 17
`edr_explore`, 8
`edr_explore()`, 16, 22
`edr_item` (`edr_items`), 10
`edr_item()`, 11
`edr_items`, 10
`edr_items()`, 11
`edr_landing`, 11
`edr_location`, 12
`edr_location()`, 3, 9, 17
`edr_locations`, 13
`edr_locations()`, 8, 10, 14, 21, 24
`edr_map`, 14
`edr_map()`, 8, 9, 22
`edr_parameter_groups`, 16
`edr_parameters`, 17
`edr_parameters()`, 9, 20
`edr_plot`, 17
`edr_plot()`, 9
`edr_position`, 19
`edr_position()`, 9
`edr_queryables`, 20
`edr_radius`, 20
`edr_request`, 21
`edr_request()`, 21
`edr_save_html`, 22
`edr_save_html()`, 9, 16
`edr_trajectory`, 23

`geojson_to_sf`, 24
`ggplot2::ggplot()`, 17

`htmlwidgets::saveWidget()`, 22

`leaflet::leaflet`, 14