Package ‘bunddev’

February 8, 2026

Title Discover and Call 'Bund.dev' APIs

Version 0.1.0

Author Michael Bücker [aut, cre]

Maintainer Michael Bücker <michael.buecker@fh-muenster.de>

Depends R (>= 4.1.0)

Description Provides a registry of APIs listed on <https://bund.dev> and a core 'OpenAPI' client layer to explore specs and perform requests. Adapter helpers return tidy tibbles for supported APIs, with optional response caching and rate limiting guidance.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

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

Config/testthat/edition 3

Imports cli, dplyr, httr2, jsonlite, purrr, rlang, stringr, tibble, tidyr, tools, xml2, yaml

VignetteBuilder knitr, rmarkdown

NeedsCompilation no

Repository CRAN

Date/Publication 2026-02-08 16:50:02 UTC

Contents

`abfallnavi_fraktionen` *List waste fractions*

Description

List waste fractions

Usage

```
abfallnavi_fraktionen(safe = TRUE, refresh = FALSE)
```

Arguments

`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Value

A tibble with waste fractions.

abfallnavi_fraktionen_hausnummern

List waste fractions for a house number

Description

List waste fractions for a house number

Usage

```
abfallnavi_fraktionen_hausnummern(hausnummern_id, safe = TRUE, refresh = FALSE)
```

Arguments

hausnummern_id	House number id.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Value

A tibble with waste fractions.

abfallnavi_fraktionen_strassen

List waste fractions for a street

Description

List waste fractions for a street

Usage

```
abfallnavi_fraktionen_strassen(strassen_id, safe = TRUE, refresh = FALSE)
```

Arguments

strassen_id	Street id.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Value

A tibble with waste fractions.

`abfallnavi_hausnummern` *List house numbers for a street*

Description

List house numbers for a street

Usage

```
abfallnavi_hausnummern(strassen_id, safe = TRUE, refresh = FALSE)
```

Arguments

`strassen_id`	Street id.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Value

A tibble with house numbers.

`abfallnavi_kalender_download` *Download calendar file*

Description

Download calendar file

Usage

```
abfallnavi_kalender_download(
  region,
  format,
  jahr,
  ort,
  strasse,
  hnr,
  fraktion,
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

region	Region code.
format	File format.
jahr	Year.
ort	Place name.
strasse	Street id.
hnr	House number id.
fraktion	Fraction ids.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Downloads a calendar file for the requested address and fraction.

Value

A tibble with raw file bytes.

---

abfallnavi_ort *Get a place by id*

Description

Get a place by id

Usage

```
abfallnavi_ort(ort_id, safe = TRUE, refresh = FALSE)
```

Arguments

ort_id	Place id.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Value

A tibble with place metadata.

`abfallnavi_orte` *List available places*

Description

List available places

Usage

```
abfallnavi_orte(safe = TRUE, refresh = FALSE)
```

Arguments

`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

The Abfallnavi API provides waste collection data for supported regions. Start by listing places (Orte) to obtain an `ortId`. Official docs: https://bundesapi.github.io/abfallnavi-api/.

Value

A tibble with places.

See Also

`abfallnavi_strassen()` for streets in a place.

Examples

```
## Not run:
abfallnavi_orte()

## End(Not run)
```

abfallnavi_strassen *List streets for a place*

Description

List streets for a place

Usage

```
abfallnavi_strassen(ort_id, safe = TRUE, refresh = FALSE)
```

Arguments

ort_id	Place id.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Value

A tibble with streets.

abfallnavi_termine_hausnummern *List collection dates for a house number*

Description

List collection dates for a house number

Usage

```
abfallnavi_termine_hausnummern(
  hausnummern_id,
  fraktion,
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

hausnummern_id	House number id.
fraktion	Fraction ids.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Value

A tibble with collection dates.

Includes `date_time` as POSIXct in Europe/Berlin.

`abfallnavi_termine_strassen` *List collection dates for a street*

Description

List collection dates for a street

Usage

```
abfallnavi_termine_strassen(
  strassen_id,
  fraktion,
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

`strassen_id`	Street id.
`fraktion`	Fraction ids.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Value

A tibble with collection dates.

Includes `date_time` as POSIXct in Europe/Berlin.

`ausbildungssuche_details` *Get training offer details*

Description

Get training offer details

Usage

```
ausbildungssuche_details(offer_id, safe = TRUE, refresh = FALSE)
```

Arguments

`offer_id`	Offer id.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns detailed information for a single offer.

Value

A tibble with offer details.

See Also

`ausbildungssuche_search()` to find offer ids.

Examples

```
## Not run:
Sys.setenv(AUSBILDUNGSSUCHE_API_KEY = "infosysbub-absuche")
bunddev_auth_set("ausbildungssuche", type = "api_key", env_var = "AUSBILDUNGSSUCHE_API_KEY")
ausbildungssuche_details(12345)

## End(Not run)
```

`ausbildungssuche_search` *Search training offers*

Description

Search training offers

Usage

```
ausbildungssuche_search(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`params`	Query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

The Ausbildungssuche API provides training offer data from the Bundesagentur fuer Arbeit. Authentication is required via `X-API-Key` (clientId `infosysbub-absuche`, discoverable from https://web.arbeitsagentur.de/weite
Official docs: https://bundesapi.github.io/ausbildungssuche-api/.

This adapter uses the `X-API-Key` header. Set it via `bunddev_auth_set()` and `AUSBILDUNGSSUCHE_API_KEY`.

Value

A tibble with training offers.

Includes `aktualisierungsdatum_time` as POSIXct in Europe/Berlin.

See Also

`ausbildungssuche_details()` for a single offer and `bunddev_auth_set()` for authentication.

Examples

```
## Not run:
Sys.setenv(AUSBILDUNGSSUCHE_API_KEY = "infosysbub-absuche")
bunddev_auth_set("ausbildungssuche", type = "api_key", env_var = "AUSBILDUNGSSUCHE_API_KEY")
ausbildungssuche_search(params = list(size = 5))

## End(Not run)
```

---

autobahn_charging_stations

List Autobahn charging stations

Description

List Autobahn charging stations

Usage

```
autobahn_charging_stations(road_id, flatten = FALSE, flatten_mode = "json")
```

Arguments

road_id	Road identifier.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns charging stations for a specific Autobahn road id. Official docs: https://autobahn.api.bund.dev.

Value

A tibble with charging stations.

See Also

autobahn_roads() and autobahn_charging_station_details().

Examples

```
## Not run:
roads <- autobahn_roads()
autobahn_charging_stations(roads$road_id[[1]], flatten = TRUE)

## End(Not run)
```

`autobahn_charging_station_details` *Get Autobahn charging station details*

Description

Get Autobahn charging station details

Usage

```
autobahn_charging_station_details(
  station_id,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`station_id`	Charging station identifier.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns full details for a single charging station entry. Official docs: https://autobahn.api.bund.dev.

Value

A tibble with charging station details.

See Also

`autobahn_charging_stations()` to list stations.

Examples

```
## Not run:
roads <- autobahn_roads()
stations <- autobahn_charging_stations(roads$road_id[[1]])
autobahn_charging_station_details(stations$identifier[[1]])

## End(Not run)
```

`autobahn_closures` *List Autobahn closures*

Description

List Autobahn closures

Usage

```
autobahn_closures(road_id, flatten = FALSE, flatten_mode = "json")
```

Arguments

`road_id`	Road identifier.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns current closures for a specific Autobahn road id. Official docs: https://autobahn.api.bund.dev.

Value

A tibble with closures.

See Also

`autobahn_roads()` and `autobahn_closure_details()`.

Examples

```
## Not run:
roads <- autobahn_roads()
autobahn_closures(roads$road_id[[1]], flatten = TRUE)

## End(Not run)
```

`autobahn_closure_details` *Get Autobahn closure details*

Description

Get Autobahn closure details

Usage

```
autobahn_closure_details(closure_id, flatten = FALSE, flatten_mode = "json")
```

Arguments

`closure_id`	Closure identifier.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns full details for a single closure entry. Official docs: https://autobahn.api.bund.dev.

Value

A tibble with closure details.

See Also

`autobahn_closures()` to list closures.

Examples

```
## Not run:
roads <- autobahn_roads()
closures <- autobahn_closures(roads$road_id[[1]])
autobahn_closure_details(closures$identifier[[1]])

## End(Not run)
```

`autobahn_parking_lorries` *List Autobahn lorry parking areas*

Description

List Autobahn lorry parking areas

Usage

```
autobahn_parking_lorries(road_id, flatten = FALSE, flatten_mode = "json")
```

Arguments

`road_id`	Road identifier.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns lorry parking areas for a specific Autobahn road id. Official docs: https://autobahn.api.bund.dev.

Value

A tibble with lorry parking areas.

See Also

`autobahn_roads()` and `autobahn_parking_lorry_details()`.

Examples

```
## Not run:
roads <- autobahn_roads()
autobahn_parking_lorries(roads$road_id[[1]], flatten = TRUE)

## End(Not run)
```

`autobahn_parking_lorry_details` *Get Autobahn lorry parking details*

Description

Get Autobahn lorry parking details

Usage

```
autobahn_parking_lorry_details(
  lorry_id,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`lorry_id`	Lorry parking identifier.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns full details for a single lorry parking entry. Official docs: https://autobahn.api.bund.dev.

Value

A tibble with lorry parking details.

See Also

`autobahn_parking_lorries()` to list parking areas.

Examples

```
## Not run:
roads <- autobahn_roads()
parking <- autobahn_parking_lorries(roads$road_id[[1]])
autobahn_parking_lorry_details(parking$identifier[[1]])

## End(Not run)
```

autobahn_roads *List Autobahn road ids*

Description

List Autobahn road ids

Usage

```
autobahn_roads()
```

Details

Lists Autobahn road ids from the Autobahn App API (Autobahn GmbH). Official docs: https://autobahn.api.bund.dev.

Value

A tibble with available road ids.

See Also

`autobahn_roadworks()` and `autobahn_warnings()` for road-specific data.

Examples

```
## Not run:
autobahn_roads()

## End(Not run)
```

autobahn_roadworks *List Autobahn roadworks*

Description

List Autobahn roadworks

Usage

```
autobahn_roadworks(road_id, flatten = FALSE, flatten_mode = "json")
```

Arguments

road_id	Road identifier.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns current roadworks for a specific Autobahn road id. Official docs: https://autobahn.api.bund.dev.

Value

A tibble with roadworks.

See Also

`autobahn_roads()` to list available roads, and `autobahn_roadwork_details()` for detail records.

Examples

```
## Not run:
roads <- autobahn_roads()
autobahn_roadworks(roads$road_id[[1]], flatten = TRUE)

## End(Not run)
```

`autobahn_roadwork_details` *Get Autobahn roadwork details*

Description

Get Autobahn roadwork details

Usage

```
autobahn_roadwork_details(roadwork_id, flatten = FALSE, flatten_mode = "json")
```

Arguments

`roadwork_id`	Roadwork identifier.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns full details for a single roadwork entry. Official docs: https://autobahn.api.bund.dev.

Value

A tibble with roadwork details.

See Also

autobahn_roadworks() to list roadworks.

Examples

```
## Not run:
roads <- autobahn_roads()
roadworks <- autobahn_roadworks(roads$road_id[[1]])
autobahn_roadwork_details(roadworks$identifier[[1]])

## End(Not run)
```

---

autobahn_warnings *List Autobahn warnings*

Description

List Autobahn warnings

Usage

```
autobahn_warnings(road_id, flatten = FALSE, flatten_mode = "json")
```

Arguments

road_id	Road identifier.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns current warnings for a specific Autobahn road id. Official docs: https://autobahn.api.bund.dev.

Value

A tibble with warnings.

See Also

autobahn_roads() to list roads and autobahn_warning_details() for details.

Examples

```
## Not run:
roads <- autobahn_roads()
autobahn_warnings(roads$road_id[[1]], flatten = TRUE)

## End(Not run)
```

`autobahn_warning_details` *Get Autobahn warning details*

Description

Get Autobahn warning details

Usage

```
autobahn_warning_details(warning_id, flatten = FALSE, flatten_mode = "json")
```

Arguments

`warning_id`	Warning identifier.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns full details for a single warning entry. Official docs: https://autobahn.api.bund.dev.

Value

A tibble with warning details.

See Also

`autobahn_warnings()` to list warnings.

Examples

```
## Not run:
roads <- autobahn_roads()
warnings <- autobahn_warnings(roads$road_id[[1]])
autobahn_warning_details(warnings$identifier[[1]])

## End(Not run)
```

`autobahn_webcams` *List Autobahn webcams*

Description

List Autobahn webcams

Usage

```
autobahn_webcams(road_id, flatten = FALSE, flatten_mode = "json")
```

Arguments

`road_id`	Road identifier.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns webcam entries for a specific Autobahn road id. Official docs: https://autobahn.api.bund.dev.

Value

A tibble with webcams.

See Also

`autobahn_roads()` and `autobahn_webcam_details()`.

Examples

```
## Not run:
roads <- autobahn_roads()
autobahn_webcams(roads$road_id[[1]], flatten = TRUE)

## End(Not run)
```

`autobahn_webcam_details` *Get Autobahn webcam details*

Description

Get Autobahn webcam details

Usage

```
autobahn_webcam_details(webcam_id, flatten = FALSE, flatten_mode = "json")
```

Arguments

`webcam_id`	Webcam identifier.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns full details for a single webcam entry. Official docs: https://autobahn.api.bund.dev.

Value

A tibble with webcam details.

See Also

`autobahn_webcams()` to list webcams.

Examples

```
## Not run:
roads <- autobahn_roads()
webcams <- autobahn_webcams(roads$road_id[[1]])
autobahn_webcam_details(webcams$identifier[[1]])

## End(Not run)
```

`berufssprachkurssuche_search` *Search language course offers*

Description

Search language course offers

Usage

```
berufssprachkurssuche_search(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`params`	Query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

The Berufssprachkurssuche API provides language course data. Authentication can be done via OAuth2 client credentials or by sending the public client id as `X-API-Key` (clientId `bd24f42e-ad0b-4005-b834-23bb6800dc`). Official docs: https://berufssprachkurssuche.api.bund.dev/.

Recommended: Configure OAuth2 via `bunddev_auth_set()` with type `"oauth2"`. If you provide a client secret, it fetches an OAuth token; otherwise it falls back to sending the client ID as `X-API-Key`.

Value

A tibble with course offers.

Includes `beginn_time`, `ende_time`, and `anmeldeschluss_time` as POSIXct in Europe/Berlin.

See Also

`bunddev_auth_set()` to configure authentication.

Examples

```
## Not run:
# Recommended: OAuth2 configuration
Sys.setenv(
  BERUFSSPRACHKURSSUCHE_API_KEY = "bd24f42e-ad0b-4005-b834-23bb6800dc6c",
  BERUFSSPRACHKURSSUCHE_CLIENT_SECRET = "<client-secret>"
)
bunddev_auth_set("berufssprachkurssuche",
  type = "oauth2",
  oauth_url = "https://rest.arbeitsagentur.de/oauth/gettoken_cc",
  env_var = "BERUFSSPRACHKURSSUCHE_API_KEY",
  oauth_secret_env = "BERUFSSPRACHKURSSUCHE_CLIENT_SECRET",
  oauth_default_id = "bd24f42e-ad0b-4005-b834-23bb6800dc6c")
berufssprachkurssuche_search(params = list(page = 0, systematiken = "MC"))

## End(Not run)
```

---

bewerberboerse_details

Retrieve Bewerberboerse candidate details

Description

Retrieve Bewerberboerse candidate details

Usage

```
bewerberboerse_details(referenznummer, flatten = FALSE, flatten_mode = "json")
```

Arguments

`referenznummer`	Bewerber referenznummer.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Fetches details for a single candidate. The `referenznummer` typically comes from `bewerberboerse_search()`. See https://bundesapi.github.io/bewerberboerse-api/.

Value

A tibble containing detailed information for a single candidate, including personal details, skills, work history, education, and contact preferences. Structure is similar to `bewerberboerse_search()` results.

See Also

bewerberboerse_search() to find candidates and bunddev_auth_set() for auth.

Examples

```
## Not run:
Sys.setenv(BEWERBERBOERSE_API_KEY = "jobboerse-bewerbersuche-ui")
bunddev_auth_set("bewerberboerse", type = "api_key", env_var = "BEWERBERBOERSE_API_KEY")
bewerberboerse_details("12345", flatten = TRUE)

## End(Not run)
```

---

bewerberboerse_search *Search the Bewerberboerse API*

Description

Search the Bewerberboerse API

Usage

```
bewerberboerse_search(params = list(), flatten = FALSE, flatten_mode = "json")
```

Arguments

params	List of query parameters.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

The Bewerberboerse API provides access to candidate listings. Authentication uses an API key passed as X-API-Key (clientId jobboerse-bewerbersuche-ui). See https://bundesapi.github.io/bewerberboerse-api/ for official docs.

Use bunddev_auth_set() to configure the key and bunddev_parameters() to inspect available query parameters.

Value

A tibble containing candidate listings with columns including reference number, availability, location, skills, and contact flags. When flatten = FALSE, includes nested list columns for education and experience. Metadata columns include page, size, and max_ergebnisse.

See Also

bewerberboerse_details() for detailed entries and bunddev_auth_set() for authentication setup.

Examples

```
## Not run:
Sys.setenv(BEWERBERBOERSE_API_KEY = "jobboerse-bewerbersuche-ui")
bunddev_auth_set("bewerberboerse", type = "api_key", env_var = "BEWERBERBOERSE_API_KEY")
bewerberboerse_search(params = list(was = "data", size = 10), flatten = TRUE)

## End(Not run)
```

`bunddev_auth_get` *Get authentication configuration for an API*

Description

Get authentication configuration for an API

Usage

```
bunddev_auth_get(api)
```

Arguments

`api`	Registry id.

Details

Returns the stored auth configuration for the API or a default none entry if no auth has been configured.

Value

A list with auth settings.

See Also

`bunddev_auth_set()` to configure credentials.

Examples

```
bunddev_auth_get("jobsuche")
```

bunddev_auth_set *Set authentication configuration for an API*

Description

Set authentication configuration for an API

Usage

```
bunddev_auth_set(
  api,
  type = "api_key",
  env_var,
  scheme = NULL,
  oauth_url = NULL,
  oauth_secret_env = NULL,
  oauth_default_id = NULL,
  oauth_token_header = "OAuthAccessToken",
  oauth_fallback_header = "X-API-Key"
)
```

Arguments

api	Registry id.
type	Authentication type ("none", "api_key", or "oauth2").
env_var	Environment variable containing credentials (for api_key) or client ID (for oauth2).
scheme	Authentication scheme used in the Authorization header (e.g., "Bearer", "ApiKey", "OAuth").
oauth_url	OAuth2 token endpoint URL (for oauth2 type).
oauth_secret_env	OAuth2 client secret environment variable (for oauth2 type).
oauth_default_id	Default OAuth2 client ID if env_var is not set (for oauth2 type).
oauth_token_header	Header name for OAuth token (default "OAuthAccessToken").
oauth_fallback_header	Header name when no secret available (default "X-API-Key").

Details

Store auth configuration used by bunddev_call() and adapter helpers.

For API key auth: set env_var to the name of an environment variable containing the key. The key is sent as Authorization: {scheme} <key>.

For OAuth2 client credentials: set oauth_url to the token endpoint, env_var to the client ID env var, and oauth_secret_env to the client secret env var. If the secret is available, fetches an OAuth token; otherwise falls back to sending the client ID as an API key.

Value

The updated auth configuration.

See Also

bunddev_auth_get() to inspect the stored configuration, and bunddev_call() to make authenticated requests.

Examples

```
# API key authentication
Sys.setenv(JOBBOERSE_API_KEY = "jobboerse-jobsuche")
bunddev_auth_set("jobsuche", type = "api_key", env_var = "JOBBOERSE_API_KEY", scheme = "X-API-Key")

# OAuth2 client credentials
bunddev_auth_set("berufssprachkurssuche",
  type = "oauth2",
  oauth_url = "https://rest.arbeitsagentur.de/oauth/gettoken_cc",
  env_var = "BERUFSSPRACHKURSSUCHE_API_KEY",
  oauth_secret_env = "BERUFSSPRACHKURSSUCHE_CLIENT_SECRET",
  oauth_default_id = "bd24f42e-ad0b-4005-b834-23bb6800dc6c")
```

---

bunddev_cache_dir *Locate the bunddev cache directory*

Description

Locate the bunddev cache directory

Usage

```
bunddev_cache_dir()
```

Details

The cache directory is used to store downloaded OpenAPI specs and cached API responses. Use this to inspect or clean cached files.

Value

Cache directory path.

See Also

bunddev_spec() to download specs, and bunddev_spec_path() to locate a specific spec file.

Examples

```
bunddev_cache_dir()
```

`bunddev_call` *Call an API operation*

Description

Call an API operation

Usage

```
bunddev_call(
  api,
  operation_id = NULL,
  params = list(),
  path = NULL,
  method = NULL,
  parse = c("json", "text", "raw", "xml"),
  base_url = NULL,
  body = NULL,
  body_type = c("json", "form"),
  headers = NULL,
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

`api`	Registry id.
`operation_id`	OpenAPI operationId (use this OR path+method).
`params`	List of query parameters.
`path`	API path (use with method instead of operation_id).
`method`	HTTP method (use with path instead of operation_id).
`parse`	Response parsing mode.
`base_url`	Optional base URL override.
`body`	Optional request body (for POST/PUT requests).
`body_type`	Body encoding type ("json" or "form").
`headers`	Optional named list of custom HTTP headers.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached GET responses.

Details

This is the low-level OpenAPI caller. It supports two modes:

- Use `operation_id` to lookup endpoints by their OpenAPI operationId
- Use `path` + `method` for APIs without operationIds

The function fills path parameters from `params`, applies auth (if configured), and optionally caches GET responses when `safe = TRUE`.

Use `bunddev_parameters()` and `bunddev_parameter_values()` to discover valid parameters before calling.

Value

Parsed response.

See Also

`bunddev_call_tidy()` for adapter-specific tidy outputs, and `bunddev_auth_set()` to configure API keys.

Examples

```
# Retrieve Autobahn road ids (raw response)
bunddev_call("autobahn", "list-autobahnen")
```

---

bunddev_call_tidy *Call an API operation and return tidy data*

Description

Call an API operation and return tidy data

Usage

```
bunddev_call_tidy(api, operation_id, params = list(), ...)
```

Arguments

`api`	Registry id.
`operation_id`	OpenAPI operationId.
`params`	List of parameters.
`...`	Additional arguments passed to the tidier.

Details

This helper calls `bunddev_call()` and then applies the adapter-specific tidier for the API. If no tidier is registered, a tibble containing the raw response is returned.

Value

A tibble with tidied results.

See Also

bunddev_call() for raw responses, and bunddev_parameters() to discover available parameters.

Examples

```
# Tidy Autobahn roadworks
bunddev_call_tidy("autobahn", "list-roadworks", params = list(roadId = "A1"))
```

---

bunddev_endpoints *Retrieve endpoints for a given API*

Description

Retrieve endpoints for a given API

List OpenAPI endpoints for an API

Usage

```
bunddev_endpoints(id)

bunddev_endpoints(id)
```

Arguments

id	Registry id.

Details

Parses the cached OpenAPI spec and returns every available path + method with its operationId and summary, if provided.

Value

A tibble with columns method, path, operation_id, summary

A tibble with endpoints.

See Also

bunddev_parameters() to inspect parameters and bunddev_spec() to access the full spec.

Examples

```
bunddev_endpoints("autobahn")
```

bunddev_info *Get a registry entry by id*

Description

Get a registry entry by id

Usage

```
bunddev_info(id)
```

Arguments

id	Registry id.

Details

Use this to access the spec URL, documentation URL, and authentication requirements for a single API.

Value

A tibble with a single registry entry.

See Also

bunddev_list() for discovery and bunddev_registry() for the full table.

Examples

```
bunddev_info("smard")
```

bunddev_list *List registry entries*

Description

List registry entries

Usage

```
bunddev_list(tag = NULL, auth = NULL)
```

Arguments

tag	Optional tag to filter on.
auth	Optional auth type to filter on.

Details

Use this to quickly narrow down APIs by topic or authentication type. Tags correspond to the taxonomy in the bundled registry.

Value

A tibble of registry entries.

See Also

bunddev_registry() for the full table and bunddev_info() for one entry.

Examples

```
bunddev_list(tag = "jobs")
bunddev_list(auth = "api_key")
```

---

bunddev_ms_to_posix *Convert milliseconds to POSIXct*

Description

Convert milliseconds to POSIXct

Usage

```
bunddev_ms_to_posix(value, tz = "Europe/Berlin")
```

Arguments

value	Timestamp in milliseconds.
tz	Timezone for conversion.

Details

Converts epoch milliseconds to POSIXct in the requested timezone.

Value

POSIXct timestamp.

Examples

```
bunddev_ms_to_posix(1704067200000, tz = "Europe/Berlin")
```

`bunddev_parameters` *List OpenAPI parameters for an API*

Description

List OpenAPI parameters for an API

Usage

```
bunddev_parameters(id, name = NULL, path = NULL, method = NULL)
```

Arguments

`id`	Registry id.
`name`	Optional parameter name to filter.
`path`	Optional endpoint path to filter.
`method`	Optional HTTP method to filter.

Details

Returns one row per parameter defined in the OpenAPI spec. Enumerations are stored in a list-column (`enum`). Use the filter arguments to narrow down results to a specific endpoint or parameter name.

Value

A tibble with parameter metadata.

See Also

`bunddev_parameters_for()` for adapter-specific parameters and `bunddev_parameter_values()` for enum values.

Examples

```
bunddev_parameters("smard", name = "resolution")
```

bunddev_parameters_for

List OpenAPI parameters for a specific adapter

Description

List OpenAPI parameters for a specific adapter

Usage

```
bunddev_parameters_for(endpoint)
```

Arguments

endpoint	Adapter function or its name.

Details

Resolves the adapter to an OpenAPI path/method mapping and filters parameters accordingly.

Value

A tibble with parameter metadata.

See Also

bunddev_parameters() for the full API parameter table.

Examples

```
bunddev_parameters_for(smard_timeseries)
```

bunddev_parameter_values

Extract parameter enum values

Description

Extract parameter enum values

Usage

```
bunddev_parameter_values(endpoint, name)
```

Arguments

endpoint	Adapter function or its name.
name	Parameter name.

Details

Returns unique enum values for a parameter defined on the adapter endpoint.

Value

A character vector of enum values.

See Also

bunddev_parameters_for() to inspect all parameters for an adapter.

Examples

```
bunddev_parameter_values(smard_timeseries, "resolution")
```

---

bunddev_rate_limit_get

Get API rate limit configuration

Description

Get API rate limit configuration

Usage

```
bunddev_rate_limit_get(api)
```

Arguments

api	Registry id.

Details

If no explicit limit was set, the function tries to infer one from the registry entry. The result is used by adapter helpers when safe = TRUE.

Value

The rate limit configuration.

See Also

bunddev_rate_limit_set() to override the default.

Examples

```
bunddev_rate_limit_get("smard")
```

---

bunddev_rate_limit_set *Set API rate limit configuration*

Description

Set API rate limit configuration

Usage

```
bunddev_rate_limit_set(api, max_per_hour)
```

Arguments

api	Registry id.
max_per_hour	Maximum number of calls per hour.

Details

Use this to override or enforce a per-hour rate limit for a given API. The default is inferred from the registry entry when available.

Value

The stored rate limit configuration.

See Also

bunddev_rate_limit_get() to inspect the current setting.

Examples

```
bunddev_rate_limit_set("smard", max_per_hour = 60)
```

bunddev_registry *Read the bundled API registry*

Description

Read the bundled API registry

Usage

```
bunddev_registry()
```

Details

The registry is bundled with the package and contains metadata such as the API title, provider, documentation URL, OpenAPI spec URL, authentication type, and any declared rate limits. The data originates from bund.dev and the bundesAPI registry.

Value

A tibble with registry entries.

See Also

`bunddev_list()` for filtered listings, `bunddev_search()` for keyword searches, and `bunddev_info()` for a single entry.

Examples

```
registry <- bunddev_registry()
head(registry, 3)
```

bunddev_search *Search registry entries*

Description

Search registry entries

Usage

```
bunddev_search(q)
```

Arguments

q	Search query.

Details

Searches across registry ids, titles, providers, and tags using a simple substring match.

Value

A tibble of matching registry entries.

See Also

`bunddev_list()` to filter by tag or auth, and `bunddev_info()` for details on a single API.

Examples

```
bunddev_search("weather")
```

---

bunddev_spec *Retrieve a cached API spec*

Description

Retrieve a cached API spec

Usage

```
bunddev_spec(id, refresh = FALSE)
```

Arguments

`id`	Registry id.
`refresh`	Logical; refresh cached spec.

Details

Downloads the OpenAPI spec from the registry if it is missing or when `refresh = TRUE`. Parsed specs are returned as lists.

Value

Parsed OpenAPI spec.

See Also

`bunddev_endpoints()` to list operations and `bunddev_parameters()` to inspect parameters.

Examples

```
bunddev_spec("smard")
```

bunddev_spec_path *Build a cache path for an API spec*

Description

Build a cache path for an API spec

Usage

```
bunddev_spec_path(id)
```

Arguments

id	Registry id.

Details

Determines the cache file name based on the spec URL extension (.yaml/.json).

Value

File path for the cached spec.

See Also

bunddev_spec() to download and parse specs.

Examples

```
bunddev_spec_path("smard")
```

bunddev_timestamp_to_ms *Convert timestamps to milliseconds*

Description

Convert timestamps to milliseconds

Usage

```
bunddev_timestamp_to_ms(value, tz = "Europe/Berlin")
```

Arguments

value	Timestamp as numeric (ms), Date, or POSIXct.
tz	Timezone for Date/POSIXct conversion.

Details

This helper standardizes timestamps for APIs that expect epoch milliseconds. Dates are interpreted in the supplied timezone.

Value

Numeric timestamp in milliseconds.

Examples

```
bunddev_timestamp_to_ms(as.POSIXct("2024-01-01 00:00:00", tz = "Europe/Berlin"))
```

---

bundeshaushalt_budget_data

Query Bundeshaushalt budget data

Description

Query Bundeshaushalt budget data

Usage

```
bundeshaushalt_budget_data(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

The Bundeshaushalt API provides budget data for federal income and expenses. Required query parameters are `year` and `account`. Official docs: https://github.com/bundesAPI/bundeshaushalt-api.

Value

A tibble with budget data and nested detail lists.

Includes `timestamp_time` as POSIXct in Europe/Berlin.

Examples

```
## Not run:
bundeshaushalt_budget_data(params = list(year = 2021, account = "expenses"))

## End(Not run)
```

---

bundesrat_aktuelles *List current Bundesrat news*

Description

List current Bundesrat news

Usage

```
bundesrat_aktuelles(view = "renderXml", safe = TRUE, refresh = FALSE)
```

Arguments

`view`	Rendering mode for the XML output.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns current news items from the Bundesrat app feed. Official docs: https://bundesrat.api.bund.dev.

Value

A tibble with news items.

Examples

```
## Not run:
bundesrat_aktuelles()

## End(Not run)
```

`bundesrat_mitglieder` *List Bundesrat members*

Description

List Bundesrat members

Usage

```
bundesrat_mitglieder(view = "renderXml", safe = TRUE, refresh = FALSE)
```

Arguments

`view`	Rendering mode for the XML output.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns Bundesrat member entries from the mobile feed. Official docs: https://bundesrat.api.bund.dev.

Value

A tibble with member entries.

Examples

```
## Not run:
bundesrat_mitglieder()

## End(Not run)
```

`bundesrat_plenum_aktuelle_sitzung` *List Bundesrat current plenum session entries*

Description

List Bundesrat current plenum session entries

Usage

```
bundesrat_plenum_aktuelle_sitzung(
  view = "renderXml",
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

`view`	Rendering mode for the XML output.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns entries for the current plenum session. Official docs: https://bundesrat.api.bund.dev.

Value

A tibble with current plenum session entries.

Examples

```
## Not run:
bundesrat_plenum_aktuelle_sitzung()

## End(Not run)
```

bundesrat_plenum_chronologisch
List Bundesrat plenum entries in chronological order

Description

List Bundesrat plenum entries in chronological order

Usage

```
bundesrat_plenum_chronologisch(
  view = "renderXml",
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

view	Rendering mode for the XML output.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns plenum entries ordered chronologically. Official docs: https://bundesrat.api.bund.dev.

Value

A tibble with plenum entries.

Examples

```
## Not run:
bundesrat_plenum_chronologisch()

## End(Not run)
```

bundesrat_plenum_kompakt

List Bundesrat plenum compact entries

Description

List Bundesrat plenum compact entries

Usage

```
bundesrat_plenum_kompakt(view = "renderXml", safe = TRUE, refresh = FALSE)
```

Arguments

view	Rendering mode for the XML output.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns plenum compact entries for Bundesrat sessions. Official docs: https://bundesrat.api.bund.dev.

Value

A tibble with plenum compact entries.

Examples

```
## Not run:
bundesrat_plenum_kompakt()

## End(Not run)
```

bundesrat_plenum_naechste_sitzungen

Get Bundesrat upcoming plenum sessions

Description

Get Bundesrat upcoming plenum sessions

Usage

```
bundesrat_plenum_naechste_sitzungen(
  view = "render[iOSDetailsWithoutInnerDate]",
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

view	Rendering mode for the XML output.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns upcoming Bundesrat sessions. Official docs: https://bundesrat.api.bund.dev.

Value

A tibble with upcoming session metadata.

Examples

```
## Not run:
bundesrat_plenum_naechste_sitzungen()

## End(Not run)
```

---

bundesrat_praesidium *List Bundesrat presidium entries*

Description

List Bundesrat presidium entries

Usage

```
bundesrat_praesidium(view = "renderXml", safe = TRUE, refresh = FALSE)
```

Arguments

view	Rendering mode for the XML output.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns Bundesrat presidium entries. Official docs: https://bundesrat.api.bund.dev.

Value

A tibble with presidium entries.

Examples

```
## Not run:
bundesrat_praesidium()

## End(Not run)
```

`bundesrat_startlist` *List Bundesrat API endpoints*

Description

List Bundesrat API endpoints

Usage

```
bundesrat_startlist(view = "renderXml", safe = TRUE, refresh = FALSE)
```

Arguments

`view`	Rendering mode for the XML output.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns the start list for the Bundesrat mobile API, including the URLs for other available resources. Official docs: https://bundesrat.api.bund.dev.

Value

A tibble with endpoint metadata.

Examples

```
## Not run:
bundesrat_startlist()

## End(Not run)
```

`bundesrat_stimmverteilung` *Get Bundesrat voting distribution*

Description

Get Bundesrat voting distribution

Usage

```
bundesrat_stimmverteilung(
  view = "render[iOSDetailsWithoutInnerDate]",
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

`view`	Rendering mode for the XML output.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns the Bundesrat voting distribution. Official docs: https://bundesrat.api.bund.dev.

Value

A tibble with voting distribution metadata.

Examples

```
## Not run:
bundesrat_stimmverteilung()

## End(Not run)
```

bundesrat_termine *List Bundesrat dates and events*

Description

List Bundesrat dates and events

Usage

```
bundesrat_termine(view = "renderXml", safe = TRUE, refresh = FALSE)
```

Arguments

view	Rendering mode for the XML output.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns scheduled Bundesrat dates and events. Official docs: https://bundesrat.api.bund.dev.

Value

A tibble with dates and events.

Examples

```
## Not run:
bundesrat_termine()

## End(Not run)
```

bundestag_article *Get a Bundestag news article*

Description

Get a Bundestag news article

Usage

```
bundestag_article(article_id, safe = TRUE, refresh = FALSE)
```

Arguments

article_id	Article id.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns a single Bundestag news article in XML format. Official docs: https://bundesapi.github.io/bundestag-api/.

Value

A tibble with article fields.

Examples

```
## Not run:
bundestag_article(849630)

## End(Not run)
```

---

bundestag_ausschuesse *List Bundestag committees*

Description

List Bundestag committees

Usage

```
bundestag_ausschuesse(safe = TRUE, refresh = FALSE)
```

Arguments

safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns the committee index from Bundestag XML feeds. Official docs: https://bundesapi.github.io/bundestag-api/.

Value

A tibble with committee entries.

Examples

```
## Not run:
bundestag_ausschuesse()

## End(Not run)
```

`bundestag_ausschuss` *Get Bundestag committee details*

Description

Get Bundestag committee details

Usage

```
bundestag_ausschuss(ausschuss_id, safe = TRUE, refresh = FALSE)
```

Arguments

`ausschuss_id`	Committee id (e.g. "a11").
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns details for a single committee. Official docs: https://bundesapi.github.io/bundestag-api/.

Value

A tibble with committee details.

Examples

```
## Not run:
bundestag_ausschuss("a11")

## End(Not run)
```

bundestag_conferences *Get Bundestag conferences overview*

Description

Get Bundestag conferences overview

Usage

```
bundestag_conferences(safe = TRUE, refresh = FALSE)
```

Arguments

safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns conference overview data from the plenum feed. Official docs: https://bundesapi.github.io/bundestag-api/.

Value

A tibble with conference fields.

Examples

```
## Not run:
bundestag_conferences()

## End(Not run)
```

bundestag_lobbyregister_search *Search the Bundestag lobbyregister*

Description

Search the Bundestag lobbyregister

Usage

```
bundestag_lobbyregister_search(
  q = NULL,
  sort = NULL,
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

q	Optional search string.
sort	Optional sorting order.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns search results from the Bundestag lobbyregister. Official docs: https://bundesapi.github.io/bundestag-lobbyregister-api/.

Value

A tibble with search metadata and result entries.

Examples

```
## Not run:
bundestag_lobbyregister_search(q = "energie")

## End(Not run)
```

---

bundestag_mdb_bio *Get a Bundestag biography*

Description

Get a Bundestag biography

Usage

```
bundestag_mdb_bio(mdb_id, safe = TRUE, refresh = FALSE)
```

Arguments

mdb_id	Member id.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns details for a single member of parliament. Official docs: https://bundesapi.github.io/bundestag-api/.

Value

A tibble with biography fields.

Examples

```
## Not run:
bundestag_mdb_bio(1769)

## End(Not run)
```

---

bundestag_mdb_index *List Bundestag members of parliament*

Description

List Bundestag members of parliament

Usage

```
bundestag_mdb_index(safe = TRUE, refresh = FALSE)
```

Arguments

safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns the index of members of parliament. Official docs: https://bundesapi.github.io/bundestag-api/.

Value

A tibble with member entries.

Examples

```
## Not run:
bundestag_mdb_index()

## End(Not run)
```

`bundestag_speaker` *Get the current Bundestag speaker*

Description

Get the current Bundestag speaker

Usage

```
bundestag_speaker(safe = TRUE, refresh = FALSE)
```

Arguments

`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns the current speaker from the plenum feed. Official docs: https://bundesapi.github.io/bundestag-api/.

Value

A tibble with speaker fields.

Examples

```
## Not run:
bundestag_speaker()

## End(Not run)
```

bundestag_video_feed *Get a Bundestag video feed entry*

Description

Get a Bundestag video feed entry

Usage

```
bundestag_video_feed(content_id, safe = TRUE, refresh = FALSE)
```

Arguments

content_id	Video content id.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns a video feed entry from the Bundestag webtv service. Official docs: https://bundesapi.github.io/bundestag-api/.

Value

A tibble with video feed fields.

Examples

```
## Not run:
bundestag_video_feed(7529016)

## End(Not run)
```

coachingangebote_search *Search coaching offers*

Description

Search coaching offers

Usage

```
coachingangebote_search(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

The Coachingangebote API provides coaching and activation offers from the Bundesagentur fuer Arbeit. Authentication can be done via OAuth2 client credentials or by sending the public client id as `X-API-Key` (clientId `ee971dcb-96fa-47b3-b2be-00863e4fc88b`). Official docs: https://bundesapi.github.io/coachingangapi/.

Recommended: Configure OAuth2 via `bunddev_auth_set()` with type `"oauth2"`. If you provide a client secret, it fetches an OAuth token; otherwise it falls back to sending the client ID as `X-API-Key`.

Value

A tibble with coaching offers.

Includes `gueltig_von_time`, `gueltig_bis_time`, and `aktualisierungsdatum_time` as POSIXct in Europe/Berlin.

See Also

`bunddev_auth_set()` to configure authentication.

Examples

```
## Not run:
# Recommended: OAuth2 configuration
Sys.setenv(
  COACHINGANGEBOTE_API_KEY = "ee971dcb-96fa-47b3-b2be-00863e4fc88b",
  COACHINGANGEBOTE_CLIENT_SECRET = "<client-secret>"
)
bunddev_auth_set("coachingangebote",
  type = "oauth2",
```

```
  oauth_url = "https://rest.arbeitsagentur.de/oauth/gettoken_cc",
  env_var = "COACHINGANGEBOTE_API_KEY",
  oauth_secret_env = "COACHINGANGEBOTE_CLIENT_SECRET",
  oauth_default_id = "ee971dcb-96fa-47b3-b2be-00863e4fc88b")
coachingangebote_search(params = list(mz = "SA 01"))

## End(Not run)
```

`dashboard_deutschland_geo` *Get Dashboard Deutschland GeoJSON*

Description

Get Dashboard Deutschland GeoJSON

Usage

```
dashboard_deutschland_geo(
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns GeoJSON data for Germany and the federal states. Official docs: https://bundesapi.github.io/dashboard-deutschland-api/.

Value

A tibble with GeoJSON metadata.

Examples

```
## Not run:
dashboard_deutschland_geo(flatten = TRUE)

## End(Not run)
```

`dashboard_deutschland_get` *List Dashboard Deutschland entries*

Description

List Dashboard Deutschland entries

Usage

```
dashboard_deutschland_get(
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns the full list of dashboard entries and metadata for each item. Official docs: https://bundesapi.github.io/dashboard-deutschland-api/.

Value

A tibble with dashboard entries.

Examples

```
## Not run:
dashboard_deutschland_get()

## End(Not run)
```

`dashboard_deutschland_indicators` *Query Dashboard Deutschland indicators*

Description

Query Dashboard Deutschland indicators

Usage

```
dashboard_deutschland_indicators(ids = NULL, safe = TRUE, refresh = FALSE)
```

Arguments

`ids`	Indicator ids, semicolon-separated or as a character vector.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns indicator data for the specified ids. Official docs: https://bundesapi.github.io/dashboard-deutschland-api/.

Value

A tibble with indicator data.

Includes `date_time` as POSIXct in Europe/Berlin.

Examples

```
## Not run:
dashboard_deutschland_indicators("tile_1667811574092")

## End(Not run)
```

`ddb_institutions` *List DDB institutions*

Description

List DDB institutions

Usage

```
ddb_institutions(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`params`	Query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns institutions registered in the DDB. Requires the DDB API key.

Value

A tibble with institution entries.

Examples

```
## Not run:
ddb_institutions(params = list(hasItems = TRUE))

## End(Not run)
```

ddb_institution_sectors *List DDB institution sectors*

Description

List DDB institution sectors

Usage

```
ddb_institution_sectors(safe = TRUE, refresh = FALSE)
```

Arguments

safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns institution sector metadata. Requires the DDB API key.

Value

A tibble with sector entries.

Examples

```
## Not run:
ddb_institution_sectors()

## End(Not run)
```

ddb_search *Search Deutsche Digitale Bibliothek*

Description

Search Deutsche Digitale Bibliothek

Usage

```
ddb_search(
  query,
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

query	Search query string.
params	Additional query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns search results from the Deutsche Digitale Bibliothek API. You need an API key from https://www.deutsche-digitale-bibliothek.de/user/apikey. The key is sent in the `Authorization` header as `OAuth oauth_consumer_key="<key>"`.

Configure authentication via `bunddev_auth_set()` using a template-style scheme, or set the `DDB_API_KEY` environment variable directly.

Value

A tibble with search metadata and result payload.

See Also

`bunddev_auth_set()` to configure authentication.

Examples

```
## Not run:
# Recommended: use bunddev_auth_set with template scheme
Sys.setenv(DDB_API_KEY = "<api-key>")
bunddev_auth_set("ddb", type = "api_key", env_var = "DDB_API_KEY",
                 scheme = "OAuth oauth_consumer_key=\"%s\"")
ddb_search(query = "berlin", params = list(rows = 5))

## End(Not run)
```

`destatis_catalogue_cubes` *List Destatis cubes*

Description

List Destatis cubes

Usage

```
destatis_catalogue_cubes(
  params = list(),
  username = "GAST",
  password = "GAST",
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

`params`	Query parameters.
`username`	Genesis username (default "GAST").
`password`	Genesis password (default "GAST").
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns the cube catalogue from the Destatis Genesis API. Official docs: https://github.com/bundesAPI/destatis-api.

Value

A tibble with the raw response payload.

Examples

```
## Not run:
destatis_catalogue_cubes()

## End(Not run)
```

`destatis_catalogue_tables` *List Destatis tables*

Description

List Destatis tables

Usage

```
destatis_catalogue_tables(
  params = list(),
  username = "GAST",
  password = "GAST",
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

`params`	Query parameters.
`username`	Genesis username (default "GAST").
`password`	Genesis password (default "GAST").
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns the table catalogue from the Destatis Genesis API. Official docs: https://github.com/bundesAPI/destatis-api.

Value

A tibble with the raw response payload.

Examples

```
## Not run:
destatis_catalogue_tables()

## End(Not run)
```

`destatis_data_cube` *Retrieve Destatis cube data*

Description

Retrieve Destatis cube data

Usage

```
destatis_data_cube(
  name,
  params = list(),
  username = "GAST",
  password = "GAST",
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

`name`	Cube id.
`params`	Query parameters.
`username`	Genesis username (default "GAST").
`password`	Genesis password (default "GAST").
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns cube data as text (csv by default). Official docs: https://github.com/bundesAPI/destatis-api.

Value

A tibble with cube data in a text column.

Examples

```
## Not run:
destatis_data_cube("21231BJ001")

## End(Not run)
```

`destatis_data_table` *Retrieve Destatis table data*

Description

Retrieve Destatis table data

Usage

```
destatis_data_table(
  name,
  params = list(),
  username = "GAST",
  password = "GAST",
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

`name`	Table id.
`params`	Query parameters.
`username`	Genesis username (default "GAST").
`password`	Genesis password (default "GAST").
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns table data as text (csv by default). Official docs: https://github.com/bundesAPI/destatis-api.

Value

A tibble with table data in a text column.

Examples

```
## Not run:
destatis_data_table("12411-0001")

## End(Not run)
```

deutschlandatlas_query *Query Deutschlandatlas indicators*

Description

Query Deutschlandatlas indicators

Usage

```
deutschlandatlas_query(
  table = "p_apo_f_ZA2022",
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

table	Table id (default "p_apo_f_ZA2022").
params	Query parameters for the ArcGIS service.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

The Deutschlandatlas API is backed by an ArcGIS feature service. You must supply a `where` filter and output format `f` (usually "json"). Official docs: https://github.com/AndreasFischer1985/deutschlandatlas-api.

Value

A tibble with indicator records.

See Also

`bunddev_parameters()` to inspect available query parameters.

Examples

```
## Not run:
deutschlandatlas_query(
  table = "p_apo_f_ZA2022",
  params = list(
    where = "1=1",
    outFields = "*",
    f = "json",
    returnGeometry = "false",
    resultRecordCount = 5
  )
)

## End(Not run)
```

`diga_catalog_entries` *List DiGA catalog entries*

Description

List DiGA catalog entries

Usage

```
diga_catalog_entries(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`params`	Query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns.

Value

A tibble containing DiGA catalog entry resources from the FHIR API. Structure depends on `flatten` setting (see `diga_device_definitions()`).

`diga_charge_item_definitions` *List DiGA prescription units*

Description

List DiGA prescription units

Usage

```
diga_charge_item_definitions(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`params`	Query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns.

Value

A tibble containing DiGA prescription unit (ChargeItemDefinition) resources from the FHIR API. Structure depends on `flatten` setting.

`diga_device_definitions` *List DiGA device definitions*

Description

List DiGA device definitions

Usage

```
diga_device_definitions(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Value

A tibble containing DiGA device definitions from the FHIR API. When `flatten = FALSE`, nested FHIR resource elements are preserved as list columns. When `flatten = TRUE`, list columns are expanded based on the specified `flatten_mode`.

Examples

```
## Not run: dig_device_definitions()
```

---

diga_organizations *List DiGA manufacturers*

Description

List DiGA manufacturers

Usage

```
diga_organizations(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns.

Value

A tibble containing DiGA manufacturer organization resources from the FHIR API. Structure depends on `flatten` setting.

---

diga_questionnaires *List DiGA questionnaires*

Description

List DiGA questionnaires

Usage

```
diga_questionnaires(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns.

Value

A tibble containing DiGA questionnaire resources from the FHIR API. Structure depends on `flatten` setting.

`diga_questionnaire_responses` *List DiGA questionnaire responses*

Description

List DiGA questionnaire responses

Usage

```
diga_questionnaire_responses(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`params`	Query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns.

Value

A tibble containing DiGA questionnaire response resources from the FHIR API. Structure depends on `flatten` setting.

`dip_bundestag_aktivitaet` *Get a DIP Aktivität*

Description

Get a DIP Aktivität

Usage

```
dip_bundestag_aktivitaet(
  aktivitaet_id,
  params = list(),
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

aktivitaet_id	Aktivität id.
params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Value

A tibble with Aktivität metadata.

Examples

```
## Not run:
dip_bundestag_aktivitaet(1493545)

## End(Not run)
```

---

dip_bundestag_aktivitaet_list *List DIP Aktivitäten*

Description

List DIP Aktivitäten

Usage

```
dip_bundestag_aktivitaet_list(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Value

A tibble with DIP response metadata.

Examples

```
## Not run:
dip_bundestag_aktivitaet_list()

## End(Not run)
```

`dip_bundestag_drucksache` *Get a DIP Drucksache*

Description

Get a DIP Drucksache

Usage

```
dip_bundestag_drucksache(
  drucksache_id,
  params = list(),
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

`drucksache_id`	Drucksache id.
`params`	Query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Value

A tibble with Drucksache metadata.

Examples

```
## Not run:
dip_bundestag_drucksache(68852)

## End(Not run)
```

dip_bundestag_drucksache_list *List DIP Drucksachen*

Description

List DIP Drucksachen

Usage

```
dip_bundestag_drucksache_list(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Value

A tibble with DIP response metadata.

Examples

```
## Not run:
dip_bundestag_drucksache_list()

## End(Not run)
```

dip_bundestag_drucksache_text

Get a DIP Drucksache text

Description

Get a DIP Drucksache text

Usage

```
dip_bundestag_drucksache_text(
  drucksache_id,
  params = list(),
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

drucksache_id	Drucksache id.
params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Value

A tibble with Drucksache text metadata.

Examples

```
## Not run:
dip_bundestag_drucksache_text(68852)

## End(Not run)
```

dip_bundestag_drucksache_text_list

List DIP Drucksache texts

Description

List DIP Drucksache texts

Usage

```
dip_bundestag_drucksache_text_list(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Value

A tibble with DIP response metadata.

Examples

```
## Not run:
dip_bundestag_drucksache_text_list()

## End(Not run)
```

---

dip_bundestag_person *Get a DIP Person*

Description

Get a DIP Person

Usage

```
dip_bundestag_person(person_id, params = list(), safe = TRUE, refresh = FALSE)
```

Arguments

person_id	Person id.
params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Value

A tibble with person metadata.

Examples

```
## Not run:
dip_bundestag_person(1728)

## End(Not run)
```

`dip_bundestag_person_list` *List DIP Personen*

Description

List DIP Personen

Usage

```
dip_bundestag_person_list(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`params`	Query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Value

A tibble with DIP response metadata.

Examples

```
## Not run:
dip_bundestag_person_list()

## End(Not run)
```

dip_bundestag_plenarprotokoll *Get a DIP Plenarprotokoll*

Description

Get a DIP Plenarprotokoll

Usage

```
dip_bundestag_plenarprotokoll(
  plenarprotokoll_id,
  params = list(),
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

plenarprotokoll_id	Plenarprotokoll id.
params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Value

A tibble with Plenarprotokoll metadata.

Examples

```
## Not run:
dip_bundestag_plenarprotokoll(908)

## End(Not run)
```

`dip_bundestag_plenarprotokoll_list` *List DIP Plenarprotokolle*

Description

List DIP Plenarprotokolle

Usage

```
dip_bundestag_plenarprotokoll_list(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`params`	Query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Value

A tibble with DIP response metadata.

Examples

```
## Not run:
dip_bundestag_plenarprotokoll_list()

## End(Not run)
```

`dip_bundestag_plenarprotokoll_text` *Get a DIP Plenarprotokoll text*

Description

Get a DIP Plenarprotokoll text

Usage

```
dip_bundestag_plenarprotokoll_text(
  plenarprotokoll_id,
  params = list(),
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

`plenarprotokoll_id`	Plenarprotokoll id.
`params`	Query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Value

A tibble with Plenarprotokoll text metadata.

Examples

```
## Not run:
dip_bundestag_plenarprotokoll_text(908)

## End(Not run)
```

`dip_bundestag_plenarprotokoll_text_list` *List DIP Plenarprotokoll texts*

Description

List DIP Plenarprotokoll texts

Usage

```
dip_bundestag_plenarprotokoll_text_list(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`params`	Query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Value

A tibble with DIP response metadata.

Examples

```
## Not run:
dip_bundestag_plenarprotokoll_text_list()

## End(Not run)
```

`dip_bundestag_vorgang` *Get a DIP Vorgang*

Description

Get a DIP Vorgang

Usage

```
dip_bundestag_vorgang(
  vorgang_id,
  params = list(),
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

`vorgang_id`	Vorgang id.
`params`	Query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns metadata for a single Vorgang.

Value

A tibble with Vorgang metadata.

Examples

```
## Not run:
dip_bundestag_vorgang(84343)

## End(Not run)
```

dip_bundestag_vorgangsposition *Get a DIP Vorgangsposition*

Description

Get a DIP Vorgangsposition

Usage

```
dip_bundestag_vorgangsposition(
  vorgangsposition_id,
  params = list(),
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

vorgangsposition_id	Vorgangsposition id.
params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Value

A tibble with Vorgangsposition metadata.

Examples

```
## Not run:
dip_bundestag_vorgangsposition(173376)

## End(Not run)
```

dip_bundestag_vorgangsposition_list
List DIP Vorgangsposition entries

Description

List DIP Vorgangsposition entries

Usage

```
dip_bundestag_vorgangsposition_list(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns metadata for Vorgangsposition entries.

Value

A tibble with DIP response metadata.

Examples

```
## Not run:
dip_bundestag_vorgangsposition_list()

## End(Not run)
```

`dip_bundestag_vorgang_list` *List DIP Vorgang entries*

Description

List DIP Vorgang entries

Usage

```
dip_bundestag_vorgang_list(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`params`	Query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns metadata for DIP Vorgang entries. Requires an API key. Obtain a key from https://dip.bundestag.de/über-dip/hilfe/api.

Configure authentication via `bunddev_auth_set()` or set the `DIP_BUNDESTAG_API_KEY` environment variable directly.

Value

A tibble with DIP response metadata.

See Also

`bunddev_auth_set()` to configure authentication.

Examples

```
## Not run:
# Recommended: use bunddev_auth_set
Sys.setenv(DIP_BUNDESTAG_API_KEY = "<api-key>")
bunddev_auth_set(
  "dip_bundestag",
  type = "api_key",
  env_var = "DIP_BUNDESTAG_API_KEY",
  scheme = "ApiKey"
)
dip_bundestag_vorgang_list()

## End(Not run)
```

---

dwd_alpine_forecast_text

Fetch DWD alpine forecast text

Description

Fetch DWD alpine forecast text

Usage

```
dwd_alpine_forecast_text(safe = TRUE, refresh = FALSE)
```

Arguments

safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Retrieves alpine forecast text from the DWD App API. Official docs: https://dwd.api.bund.dev.

Value

A tibble with the forecast text.

See Also

dwd_sea_warning_text() and dwd_avalanche_warnings().

Examples

```
## Not run:
dwd_alpine_forecast_text()

## End(Not run)
```

`dwd_avalanche_warnings` *Fetch DWD avalanche warnings*

Description

Fetch DWD avalanche warnings

Usage

```
dwd_avalanche_warnings(safe = TRUE, refresh = FALSE)
```

Arguments

`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Retrieves avalanche warnings from the DWD App API. Official docs: https://dwd.api.bund.dev.

Value

A tibble with avalanche data.

See Also

`dwd_alpine_forecast_text()` and `dwd_sea_warning_text()`.

Examples

```
## Not run:
dwd_avalanche_warnings()

## End(Not run)
```

`dwd_coast_warnings` *Fetch DWD coastal warnings*

Description

Fetch DWD coastal warnings

Usage

```
dwd_coast_warnings(
  language = c("de", "en"),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`language`	Language code ("de" or "en").
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Retrieves coastal warnings from the DWD App API. Official docs: https://dwd.api.bund.dev.

Value

A tibble with coastal warnings.

See Also

`dwd_warnings_nowcast()` and `dwd_municipality_warnings()`.

Examples

```
## Not run:
dwd_coast_warnings(language = "de", flatten = TRUE)

## End(Not run)
```

`dwd_crowd_reports` *Fetch DWD crowd reports*

Description

Fetch DWD crowd reports

Usage

```
dwd_crowd_reports(
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Retrieves crowd-sourced weather reports from the DWD App API. Official docs: https://dwd.api.bund.dev.

Value

A tibble with crowd reports.

Includes `timestamp_time` as POSIXct in Europe/Berlin.

See Also

`dwd_station_overview()` and `dwd_warnings_nowcast()`.

Examples

```
## Not run:
dwd_crowd_reports(flatten = TRUE)

## End(Not run)
```

`dwd_municipality_warnings` *Fetch DWD municipality warnings*

Description

Fetch DWD municipality warnings

Usage

```
dwd_municipality_warnings(
  language = c("de", "en"),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`language`	Language code ("de" or "en").
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Retrieves municipality warnings from the DWD App API. Official docs: https://dwd.api.bund.dev.

Value

A tibble with municipality warnings.

Includes `start_time` and `end_time` as POSIXct in Europe/Berlin.

See Also

`dwd_warnings_nowcast()` and `dwd_coast_warnings()`.

Examples

```
## Not run:
dwd_municipality_warnings(language = "de", flatten = TRUE)

## End(Not run)
```

dwd_sea_warning_text *Fetch DWD sea warning text*

Description

Fetch DWD sea warning text

Usage

```
dwd_sea_warning_text(safe = TRUE, refresh = FALSE)
```

Arguments

safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Retrieves sea warning text from the DWD App API. Official docs: https://dwd.api.bund.dev.

Value

A tibble with the warning text.

See Also

dwd_alpine_forecast_text() and dwd_avalanche_warnings().

Examples

```
## Not run:
dwd_sea_warning_text()

## End(Not run)
```

dwd_station_overview *Fetch DWD station overview data*

Description

Fetch DWD station overview data

Usage

```
dwd_station_overview(
  station_ids,
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

station_ids	Station identifiers.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Retrieves station overview data from the DWD App API. Official docs: https://dwd.api.bund.dev. Station IDs can be looked up via DWD opendata resources or the API documentation.

Value

A tibble with station data.

Includes forecast time columns (`forecast_start_time`, `forecast1_start_time`, `forecast2_start_time`) as POSIXct in Europe/Berlin.

See Also

`dwd_crowd_reports()` and `dwd_warnings_nowcast()`.

Examples

```
## Not run:
dwd_station_overview(c("10865", "G005"), flatten = TRUE)

## End(Not run)
```

`dwd_warnings_nowcast` *Fetch DWD nowcast warnings*

Description

Fetch DWD nowcast warnings

Usage

```
dwd_warnings_nowcast(
  language = c("de", "en"),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`language`	Language code ("de" or "en").
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Retrieves nowcast weather warnings from the DWD App API. Official docs: https://dwd.api.bund.dev.

Value

A tibble with nowcast warnings.

Includes `start_time` and `end_time` as POSIXct in Europe/Berlin.

See Also

`dwd_municipality_warnings()` and `dwd_coast_warnings()`.

Examples

```
## Not run:
dwd_warnings_nowcast(language = "de", flatten = TRUE)

## End(Not run)
```

`eco_visio_counters` *List Eco-Visio counters for an organization*

Description

List Eco-Visio counters for an organization

Usage

```
eco_visio_counters(id_organisme, safe = TRUE, refresh = FALSE)
```

Arguments

`id_organisme`	Organization ID (e.g., 4586 for Bike Count Display).
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

The Eco-Visio API provides access to bicycle and pedestrian counter data. This function returns all counters registered for a specific organization. Official docs: https://github.com/bundesAPI/eco-visio-api.

Value

A tibble with counter metadata.

See Also

`eco_visio_data()` to retrieve measurement data for a counter.

Examples

```
## Not run:
eco_visio_counters(4586)

## End(Not run)
```

`eco_visio_data` *Get Eco-Visio counter measurement data*

Description

Get Eco-Visio counter measurement data

Usage

```
eco_visio_data(
  id_organisme,
  id_pdc,
  interval,
  flow_ids,
  begin = NULL,
  end = NULL,
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

`id_organisme`	Organization ID.
`id_pdc`	Counter ID.
`interval`	Aggregation interval (1-6: 1=15min, 2=hours, 3=days, 4=weeks, 5=months, 6=years).
`flow_ids`	Practice IDs (semicolon-separated string or character vector).
`begin`	Optional start date (Date or "YYYY-MM-DD" string).
`end`	Optional end date (Date or "YYYY-MM-DD" string).
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns measurement data for a specific counter. The `flow_ids` parameter specifies which traffic types to include (e.g., pedestrians, cyclists). Use `eco_visio_counters()` to discover available counters and their flow IDs.

Value

A tibble with measurement data.

Includes `date_time` as POSIXct in Europe/Berlin.

See Also

`eco_visio_counters()` to list available counters.

Examples

```
## Not run:
# Get daily data for a counter
eco_visio_data(
  id_organisme = 4586,
  id_pdc = 100125331,
  interval = 4,
  flow_ids = "101125331"
)

## End(Not run)
```

`entgeltatlas_entgelte` *Query Entgeltatlas data by KldB key*

Description

Query Entgeltatlas data by KldB key

Usage

```
entgeltatlas_entgelte(
  kldb,
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`kldb`	KldB 2010 key (3-5 digits).
`params`	Query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

The Entgeltatlas API provides wage statistics for occupational categories identified by KldB keys. The API accepts optional filters for region, gender, age, sector, and performance level. Official docs: https://bundesapi.github.io/entgeltatlas-api/.

Recommended: Configure OAuth2 via `bunddev_auth_set()` with type `"oauth2"`. If you provide a client secret, it fetches an OAuth token; otherwise it falls back to sending the client ID as `X-API-Key`.

Value

A tibble with entgelt entries.

Examples

```
## Not run:
# Recommended: OAuth2 configuration
Sys.setenv(
  ENTGELTATLAS_API_KEY = "c4f0d292-9d0f-4763-87dd-d3f9e78fb006",
  ENTGELTATLAS_CLIENT_SECRET = "<client-secret>"
)
bunddev_auth_set("entgeltatlas",
  type = "oauth2",
  oauth_url = "https://rest.arbeitsagentur.de/oauth/gettoken_cc",
  env_var = "ENTGELTATLAS_API_KEY",
  oauth_secret_env = "ENTGELTATLAS_CLIENT_SECRET",
  oauth_default_id = "c4f0d292-9d0f-4763-87dd-d3f9e78fb006")
entgeltatlas_entgelte("84304", params = list(r = 1, g = 1))

## End(Not run)
```

`feiertage_list` *List German public holidays*

Description

List German public holidays

Usage

```
feiertage_list(
  jahr = NULL,
  nur_land = NULL,
  nur_daten = NULL,
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

jahr	Year to query (defaults to current year on the API).
nur_land	Optional Bundesland code to filter.
nur_daten	Logical; return only date values (1) or include names (0).
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

The Feiertage API returns German public holidays from a Wikipedia-based dataset. The endpoint supports filtering by year and Bundesland. Official docs: https://github.com/bundesAPI/feiertage-api.

Value

A tibble with holiday names and dates.

Region-level results include a region column and a note column for holiday-specific hints.

Includes date_time as POSIXct in Europe/Berlin.

See Also

bunddev_parameters() for available query parameters.

Examples

```
## Not run:
feiertage_list(jahr = 2024)
feiertage_list(jahr = 2024, nur_land = "BY")

## End(Not run)
```

---

handelsregister_search

Search the Handelsregister portal

Description

This adapter scrapes the public Handelsregister search form because no official OpenAPI specification is available for this service.

Usage

```
handelsregister_search(
  schlagwoerter,
  schlagwort_optionen = c("all", "min", "exact"),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

schlagwoerter	Search terms.
schlagwort_optionen	Keyword options.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

The Handelsregister search is provided via a public web form. This helper automates the form flow and parses the result table into a tidy tibble. Official docs: https://github.com/bundesAPI/handelsregister.

The registry notes that more than 60 requests per hour may violate the terms of use. Use `safe = TRUE` to respect the built-in rate limiting.

Value

A tibble with search results.

See Also

`bunddev_rate_limit_get()` to inspect the configured limit.

Examples

```
## Not run:
handelsregister_search("Deutsche Bahn", schlagwort_optionen = "all")

## End(Not run)
```

`hilfsmittel_nachweisschema` *Get Hilfsmittel verification schema details*

Description

Get Hilfsmittel verification schema details

Usage

```
hilfsmittel_nachweisschema(
  id,
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`id`	Nachweisschema id.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns detail information for a Nachweisschema. Official docs: https://github.com/bundesAPI/hilfsmittel-api.

Value

A tibble with Nachweisschema details.

Examples

```
## Not run:
hilfsmittel_nachweisschema("a3d37017-2c91-4d6d-bbbe-4002d2868044")

## End(Not run)
```

`hilfsmittel_produkt` *Get Hilfsmittel product details*

Description

Get Hilfsmittel product details

Usage

```
hilfsmittel_produkt(
  id,
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`id`	Produkt id.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns detail information for a single product. Official docs: https://github.com/bundesAPI/hilfsmittel-api.

Value

A tibble with product details.

Examples

```
## Not run:
hilfsmittel_produkt("f41f52a6-5d2d-4dd3-9d0e-39675ceca7f3")

## End(Not run)
```

`hilfsmittel_produktart` *Get Hilfsmittel product type details*

Description

Get Hilfsmittel product type details

Usage

```
hilfsmittel_produktart(
  id,
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`id`	Produktart id.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns details for a product type (Produktart). Official docs: https://github.com/bundesAPI/hilfsmittel-api.

Value

A tibble with product type details.

Examples

```
## Not run:
hilfsmittel_produktart("e6b913ef-cf21-4c5f-826d-f866516c3c65")

## End(Not run)
```

`hilfsmittel_produkte` *List Hilfsmittel products*

Description

List Hilfsmittel products

Usage

```
hilfsmittel_produkte(
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns the full product list (large payload). Official docs: https://github.com/bundesAPI/hilfsmittel-api.

Value

A tibble with products.

Examples

```
## Not run:
hilfsmittel_produkte()

## End(Not run)
```

`hilfsmittel_produktgruppe` *Get Hilfsmittel product group details*

Description

Get Hilfsmittel product group details

Usage

```
hilfsmittel_produktgruppe(
  id,
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`id`	Produktgruppe id.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns details for a product group (Produktgruppe). Official docs: https://github.com/bundesAPI/hilfsmittel-api.

Value

A tibble with product group details.

Examples

```
## Not run:
tree <- hilfsmittel_tree(level = 1)
hilfsmittel_produktgruppe(tree$id[[1]])

## End(Not run)
```

hilfsmittel_tree *List Hilfsmittel tree nodes*

Description

List Hilfsmittel tree nodes

Usage

```
hilfsmittel_tree(level, safe = TRUE, refresh = FALSE)
```

Arguments

level	Tree level to retrieve (1-4).
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns nodes from the Hilfsmittel product tree up to the selected level. Official docs: https://github.com/bundesAPI/hilfsmitt
api.

Value

A tibble with tree nodes.

Examples

```
## Not run:
hilfsmittel_tree(level = 1)

## End(Not run)
```

hilfsmittel_untergruppe *Get Hilfsmittel subgroup details*

Description

Get Hilfsmittel subgroup details

Usage

```
hilfsmittel_untergruppe(
  id,
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

id	Untergruppe id.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns details for a subgroup (Untergruppe). Official docs: https://github.com/bundesAPI/hilfsmittel-api.

Value

A tibble with subgroup details.

Examples

```
## Not run:
hilfsmittel_untergruppe("c92d1976-d3cb-4b9f-bcdf-805272a9ea86")

## End(Not run)
```

---

hochwasserzentralen_bundeslaender

List flood portal states and connected regions

Description

List flood portal states and connected regions

Usage

```
hochwasserzentralen_bundeslaender(safe = TRUE, refresh = FALSE)
```

Arguments

safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns metadata for all Bundeslaender and connected regions in the hochwasserzentralen.de portal. Official docs: https://bundesapi.github.io/hochwasserzentralen-api/.

Value

A tibble with Bundesland metadata.

Examples

```
## Not run:
hochwasserzentralen_bundeslaender()

## End(Not run)
```

---

hochwasserzentralen_bundesland_geojson

Get Bundesland GeoJSON boundaries

Description

Get Bundesland GeoJSON boundaries

Usage

```
hochwasserzentralen_bundesland_geojson(
  version,
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

version	GeoJSON version identifier (e.g., "20211130").
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns GeoJSON boundaries for Bundeslaender and connected regions from the hochwasserzentralen.de portal. Official docs: https://bundesapi.github.io/hochwasserzentralen-api/.

Value

A tibble with GeoJSON metadata and feature list-columns.

Examples

```
## Not run:
hochwasserzentralen_bundesland_geojson("20211130", flatten = TRUE)

## End(Not run)
```

---

hochwasserzentralen_bundesland_info

Get flood portal metadata for a Bundesland

Description

Get flood portal metadata for a Bundesland

Usage

```
hochwasserzentralen_bundesland_info(
  bundesland_id,
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

bundesland_id	Bundesland id (e.g., "HE").
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns metadata for a single Bundesland or region in the hochwasserzentralen.de portal. Official docs: https://bundesapi.github.io/hochwasserzentralen-api/.

Value

A tibble with Bundesland metadata.

Examples

```
## Not run:
hochwasserzentralen_bundesland_info("HE")

## End(Not run)
```

`hochwasserzentralen_lagepegel` *List flood gauge locations*

Description

List flood gauge locations

Usage

```
hochwasserzentralen_lagepegel(safe = TRUE, refresh = FALSE)
```

Arguments

`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns latitude/longitude coordinates for all pegel identifiers available in the hochwasserzentralen.de portal. Official docs: https://bundesapi.github.io/hochwasserzentralen-api/.

Value

A tibble with pegel coordinates.

Examples

```
## Not run:
hochwasserzentralen_lagepegel()

## End(Not run)
```

hochwasserzentralen_pegel_info *Get flood gauge information*

Description

Get flood gauge information

Usage

```
hochwasserzentralen_pegel_info(pegelnummer, safe = TRUE, refresh = FALSE)
```

Arguments

pegelnummer	Pegelnummer identifier (e.g., "HE_24820206").
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns metadata for a single flood gauge (pegel) from hochwasserzentralen.de. Official docs: https://bundesapi.github.io/hochwasserzentralen-api/.

Value

A tibble with pegel metadata.

Examples

```
## Not run:
hochwasserzentralen_pegel_info("HE_24820206")

## End(Not run)
```

interpol_red_notice *Get an Interpol red notice*

Description

Get an Interpol red notice

Usage

```
interpol_red_notice(
  notice_id,
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

notice_id	Notice ID.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns detailed data for a red notice. Official docs: https://interpol.api.bund.dev.

Value

A tibble with notice details.

Examples

```
## Not run:
interpol_red_notice("1993-27493")

## End(Not run)
```

---

interpol_red_notices *List Interpol red notices*

Description

List Interpol red notices

Usage

```
interpol_red_notices(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns Interpol red notices. Official docs: https://interpol.api.bund.dev.

Value

A tibble with notices metadata.

Examples

```
## Not run:
interpol_red_notices(params = list(resultPerPage = 10, page = 1))

## End(Not run)
```

---

interpol_red_notice_images

List Interpol red notice images

Description

List Interpol red notice images

Usage

```
interpol_red_notice_images(
  notice_id,
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

notice_id	Notice ID.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns images metadata for a red notice. Official docs: https://interpol.api.bund.dev.

Value

A tibble with image metadata.

Examples

```
## Not run:
interpol_red_notice_images("1993-27493")

## End(Not run)
```

---

interpol_un_notice *Get an Interpol UN notice*

Description

Get an Interpol UN notice

Usage

```
interpol_un_notice(
  notice_type,
  notice_id,
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

notice_type	Notice type: "persons" or "entities".
notice_id	Notice ID.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns detailed data for a UN notice. Official docs: https://interpol.api.bund.dev.

Value

A tibble with notice details.

Examples

```
## Not run:
interpol_un_notice("persons", "2021-84506")

## End(Not run)
```

---

interpol_un_notices *List Interpol UN notices*

Description

List Interpol UN notices

Usage

```
interpol_un_notices(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns UN notices from Interpol. Official docs: https://interpol.api.bund.dev.

Value

A tibble with notices metadata.

Examples

```
## Not run:
interpol_un_notices(params = list(resultPerPage = 10, page = 1))

## End(Not run)
```

---

interpol_un_notice_images

List Interpol UN notice images

Description

List Interpol UN notice images

Usage

```
interpol_un_notice_images(
  notice_type,
  notice_id,
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

notice_type	Notice type: "persons" or "entities".
notice_id	Notice ID.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns images metadata for a UN notice. Official docs: https://interpol.api.bund.dev.

Value

A tibble with image metadata.

Examples

```
## Not run:
interpol_un_notice_images("persons", "2021-84506")

## End(Not run)
```

---

interpol_yellow_notice

Get an Interpol yellow notice

Description

Get an Interpol yellow notice

Usage

```
interpol_yellow_notice(
  notice_id,
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

notice_id	Notice ID.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns detailed data for a yellow notice. Official docs: https://interpol.api.bund.dev.

Value

A tibble with notice details.

Examples

```
## Not run:
interpol_yellow_notice("2014-5590")

## End(Not run)
```

---

interpol_yellow_notices

List Interpol yellow notices

Description

List Interpol yellow notices

Usage

```
interpol_yellow_notices(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns Interpol yellow notices. Official docs: https://interpol.api.bund.dev.

Value

A tibble with notices metadata.

Examples

```
## Not run:
interpol_yellow_notices(params = list(resultPerPage = 10, page = 1))

## End(Not run)
```

---

interpol_yellow_notice_images

List Interpol yellow notice images

Description

List Interpol yellow notice images

Usage

```
interpol_yellow_notice_images(
  notice_id,
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

notice_id	Notice ID.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns images metadata for a yellow notice. Official docs: https://interpol.api.bund.dev.

Value

A tibble with image metadata.

Examples

```
## Not run:
interpol_yellow_notice_images("2014-5590")

## End(Not run)
```

---

jobsuche_logo *Fetch Jobsuche employer logo*

Description

Fetch Jobsuche employer logo

Usage

```
jobsuche_logo(hash_id, safe = TRUE, refresh = FALSE)
```

Arguments

hash_id	Logo hash id.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns the raw logo bytes for a given employer hash id. Use this together with listings returned by jobsuche_search() or jobsuche_search_app().

See https://jobsuche.api.bund.dev for API documentation.

Value

A tibble with raw logo bytes.

See Also

jobsuche_search() for listings and bunddev_auth_set() for auth setup.

Examples

```
## Not run:
Sys.setenv(JOBBOERSE_API_KEY = "jobboerse-jobsuche")
bunddev_auth_set("jobsuche", type = "api_key", env_var = "JOBBOERSE_API_KEY")
logo <- jobsuche_logo("abc123")

## End(Not run)
```

---

jobsuche_search *Search Jobsuche listings*

Description

Search Jobsuche listings

Usage

```
jobsuche_search(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

The Jobsuche API provides access to job listings from the Bundesagentur fuer Arbeit. Authentication is required via an API key passed as `X-API-Key`. See https://jobsuche.api.bund.dev for the official API documentation.

Use `bunddev_auth_set()` to configure the key and `bunddev_parameters()` to discover supported query parameters.

Value

A tibble with job listings.

Includes parsed POSIXct columns (suffix `_time`) in Europe/Berlin.

See Also

`jobsuche_search_app()` for the app endpoint, `jobsuche_logo()` to fetch employer logos, and `bunddev_auth_set()` for authentication.

Examples

```
## Not run:
Sys.setenv(JOBBOERSE_API_KEY = "jobboerse-jobsuche")
bunddev_auth_set("jobsuche", type = "api_key", env_var = "JOBBOERSE_API_KEY")
jobsuche_search(params = list(was = "data", size = 5), flatten = TRUE)

## End(Not run)
```

---

jobsuche_search_app *Search Jobsuche listings (app endpoint)*

Description

Search Jobsuche listings (app endpoint)

Usage

```
jobsuche_search_app(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

This uses the mobile app endpoint of the Jobsuche API. It shares the same authentication mechanism and parameters as `jobsuche_search()`.

See https://jobsuche.api.bund.dev for API documentation.

Value

A tibble with job listings.

Includes parsed POSIXct columns (suffix `_time`) in Europe/Berlin.

See Also

`jobsuche_search()` for the standard endpoint and `bunddev_parameters()` for parameter discovery.

Examples

```
## Not run:
Sys.setenv(JOBBOERSE_API_KEY = "jobboerse-jobsuche")
bunddev_auth_set("jobsuche", type = "api_key", env_var = "JOBBOERSE_API_KEY")
jobsuche_search_app(params = list(was = "data", size = 5), flatten = TRUE)

## End(Not run)
```

`ladestationen_query` *Query charging stations*

Description

Query charging stations

Usage

```
ladestationen_query(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`params`	Query parameters for the ArcGIS service.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

The Ladesaeulenregister API is backed by an ArcGIS feature service. You must supply a `geometry` filter and `outFields`. The ArcGIS service may require a `token` query parameter even though the API docs describe the service as public. Official docs: https://ladestationen.api.bund.dev.

Value

A tibble with charging station records.

See Also

`bunddev_parameters()` to inspect available query parameters.

Examples

```
## Not run:
geometry <- jsonlite::toJSON(
  list(
    xmin = 13.3, ymin = 52.4, xmax = 13.5, ymax = 52.6,
    spatialReference = list(wkid = 4326)
```

```
  ),
  auto_unbox = TRUE
)
ladestationen_query(params = list(
  geometry = geometry,
  geometryType = "esriGeometryEnvelope",
  where = "1=1",
  outFields = "*",
  outSR = 4326,
  f = "json",
  returnGeometry = "false",
  resultRecordCount = 5
))

## End(Not run)
```

`lebensmittelwarnung_warnings` *List food and product warnings*

Description

List food and product warnings

Usage

```
lebensmittelwarnung_warnings(
  food = list(),
  products = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`food`	Optional request options for food warnings (rows, start, sort, fq).
`products`	Optional request options for product warnings (rows, start, sort, fq).
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

The Lebensmittelwarnungen API returns German food and product safety warnings. Requests are sent as JSON to the `warnings/merged` endpoint. Official docs: https://lebensmittelwarnung.api.bund.dev and https://github.com/bundesAPI/lebensmittelwarnung-api.

The API expects an `Authorization` header. By default the adapter uses the public key documented in the OpenAPI spec. Override it with `bunddev_auth_set()` if needed.

Value

A tibble with warning entries.

Includes `published_date_time` as POSIXct in Europe/Berlin.

See Also

`bunddev_auth_set()` to configure the authorization header.

Examples

```
## Not run:
lebensmittelwarnung_warnings(food = list(rows = 10), products = list(rows = 0))

## End(Not run)
```

---

luftqualitaet_airquality

List air quality measurements

Description

List air quality measurements

Usage

```
luftqualitaet_airquality(params = list(), safe = TRUE, refresh = FALSE)
```

Arguments

`params`	Query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

The Luftqualitaet API provides air quality data and metadata from the Umweltbundesamt. Use query parameters to filter by date/time and station. Official docs: https://luftqualitaet.api.bund.dev.

Value

A tibble with air quality data.

See Also

`luftqualitaet_measures()` for measurement metadata and `luftqualitaet_components()` for pollutant components.

Examples

```
## Not run:
luftqualitaet_airquality(params = list(
  date_from = "2024-01-01",
  date_to = "2024-01-02"
))

## End(Not run)
```

---

luftqualitaet_airquality_limits

Get air quality date limits

Description

Get air quality date limits

Usage

```
luftqualitaet_airquality_limits(safe = TRUE, refresh = FALSE)
```

Arguments

`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns date limits for air quality measurements. Official docs: https://luftqualitaet.api.bund.dev.

Value

A tibble with date limits.

luftqualitaet_annualbalances

List annual balances

Description

List annual balances

Usage

```
luftqualitaet_annualbalances(params = list(), safe = TRUE, refresh = FALSE)
```

Arguments

params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns annual balance data for a component and year. Official docs: https://luftqualitaet.api.bund.dev.

Value

A tibble with annual balance data.

luftqualitaet_components

List components

Description

List components

Usage

```
luftqualitaet_components(params = list(), safe = TRUE, refresh = FALSE)
```

Arguments

params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns available pollutant components. Official docs: https://luftqualitaet.api.bund.dev.

Value

A tibble with component metadata.

`luftqualitaet_measures` *List measurements metadata*

Description

List measurements metadata

Usage

```
luftqualitaet_measures(params = list(), safe = TRUE, refresh = FALSE)
```

Arguments

`params`	Query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns measurement metadata for stations, components, and scopes. Official docs: https://luftqualitaet.api.bund.dev.

Value

A tibble with measurement metadata.

`luftqualitaet_measures_limits` *Get measurement date limits*

Description

Get measurement date limits

Usage

```
luftqualitaet_measures_limits(safe = TRUE, refresh = FALSE)
```

Arguments

safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns date limits for measurement metadata. Official docs: https://luftqualitaet.api.bund.dev.

Value

A tibble with date limits.

---

luftqualitaet_meta *List combined metadata*

Description

List combined metadata

Usage

```
luftqualitaet_meta(params = list(), safe = TRUE, refresh = FALSE)
```

Arguments

params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns combined metadata for a given use case. Official docs: https://luftqualitaet.api.bund.dev.

Value

A tibble with combined metadata.

luftqualitaet_networks *List networks*

Description

List networks

Usage

```
luftqualitaet_networks(params = list(), safe = TRUE, refresh = FALSE)
```

Arguments

params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns networks used by the Luftqualitaet API. Official docs: https://luftqualitaet.api.bund.dev.

Value

A tibble with network metadata.

luftqualitaet_scopes *List scopes*

Description

List scopes

Usage

```
luftqualitaet_scopes(params = list(), safe = TRUE, refresh = FALSE)
```

Arguments

params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns scopes used by the Luftqualitaet API. Official docs: https://luftqualitaet.api.bund.dev.

Value

A tibble with scope metadata.

`luftqualitaet_stationsettings` *List station settings*

Description

List station settings

Usage

```
luftqualitaet_stationsettings(params = list(), safe = TRUE, refresh = FALSE)
```

Arguments

`params`	Query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns station settings metadata. Official docs: https://luftqualitaet.api.bund.dev.

Value

A tibble with station settings metadata.

`luftqualitaet_stationtypes` *List station types*

Description

List station types

Usage

```
luftqualitaet_stationtypes(params = list(), safe = TRUE, refresh = FALSE)
```

Arguments

`params`	Query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns station types metadata. Official docs: https://luftqualitaet.api.bund.dev.

Value

A tibble with station types metadata.

`luftqualitaet_thresholds` *List thresholds*

Description

List thresholds

Usage

```
luftqualitaet_thresholds(params = list(), safe = TRUE, refresh = FALSE)
```

Arguments

`params`	Query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns threshold metadata for components and scopes. Official docs: https://luftqualitaet.api.bund.dev.

Value

A tibble with threshold metadata.

`luftqualitaet_transgressions` *List transgressions*

Description

List transgressions

Usage

```
luftqualitaet_transgressions(params = list(), safe = TRUE, refresh = FALSE)
```

Arguments

params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns exceedances (transgressions) data. Official docs: https://luftqualitaet.api.bund.dev.

Value

A tibble with transgressions data.

---

luftqualitaet_transgressiontypes

List transgression types

Description

List transgression types

Usage

```
luftqualitaet_transgressiontypes(params = list(), safe = TRUE, refresh = FALSE)
```

Arguments

params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns available transgression types. Official docs: https://luftqualitaet.api.bund.dev.

Value

A tibble with transgression type metadata.

marktstammdaten_filters_gaserzeugung
List MaStR filter options for gas generation

Description

List MaStR filter options for gas generation

Usage

```
marktstammdaten_filters_gaserzeugung(
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns filter definitions for public gas generation data. Official docs: https://github.com/bundesAPI/marktstammdaten-api.

Value

A tibble with filter metadata.

Examples

```
## Not run:
marktstammdaten_filters_gaserzeugung()

## End(Not run)
```

`marktstammdaten_filters_gasverbrauch` *List MaStR filter options for gas consumption*

Description

List MaStR filter options for gas consumption

Usage

```
marktstammdaten_filters_gasverbrauch(
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns filter definitions for public gas consumption data. Official docs: https://github.com/bundesAPI/marktstammdaten-api.

Value

A tibble with filter metadata.

Examples

```
## Not run:
marktstammdaten_filters_gasverbrauch()

## End(Not run)
```

`marktstammdaten_filters_stromerzeugung` *List MaStR filter options for electricity generation*

Description

List MaStR filter options for electricity generation

Usage

```
marktstammdaten_filters_stromerzeugung(
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns filter definitions for public electricity generation data. Official docs: https://github.com/bundesAPI/marktstammdaten
api.

Value

A tibble with filter metadata.

Examples

```
## Not run:
marktstammdaten_filters_stromerzeugung()

## End(Not run)
```

`marktstammdaten_filters_stromverbrauch` *List MaStR filter options for electricity consumption*

Description

List MaStR filter options for electricity consumption

Usage

```
marktstammdaten_filters_stromverbrauch(
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns filter definitions for public electricity consumption data. Official docs: https://github.com/bundesAPI/marktstammdat
api.

Value

A tibble with filter metadata.

Examples

```
## Not run:
marktstammdaten_filters_stromverbrauch()

## End(Not run)
```

`marktstammdaten_gaserzeugung` *List MaStR gas generation data*

Description

List MaStR gas generation data

Usage

```
marktstammdaten_gaserzeugung(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`params`	Query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns public gas generation data from the MaStR. Official docs: https://github.com/bundesAPI/marktstammdaten-api.

Value

A tibble with MaStR entries.

Examples

```
## Not run:
marktstammdaten_gaserzeugung(params = list(page = 1, pageSize = 5))

## End(Not run)
```

`marktstammdaten_gasverbrauch` *List MaStR gas consumption data*

Description

List MaStR gas consumption data

Usage

```
marktstammdaten_gasverbrauch(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`params`	Query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns public gas consumption data from the MaStR. Official docs: https://github.com/bundesAPI/marktstammdaten-api.

Value

A tibble with MaStR entries.

Examples

```
## Not run:
marktstammdaten_gasverbrauch(params = list(page = 1, pageSize = 5))

## End(Not run)
```

`marktstammdaten_stromerzeugung` *List MaStR electricity generation data*

Description

List MaStR electricity generation data

Usage

```
marktstammdaten_stromerzeugung(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`params`	Query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns public electricity generation data from the MaStR. Official docs: https://github.com/bundesAPI/marktstammdaten-api.

Value

A tibble with MaStR entries.

Examples

```
## Not run:
marktstammdaten_stromerzeugung(params = list(page = 1, pageSize = 5))

## End(Not run)
```

`marktstammdaten_stromverbrauch` *List MaStR electricity consumption data*

Description

List MaStR electricity consumption data

Usage

```
marktstammdaten_stromverbrauch(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`params`	Query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns public electricity consumption data from the MaStR. Official docs: https://github.com/bundesAPI/marktstammdaten-api.

Value

A tibble with MaStR entries.

Examples

```
## Not run:
marktstammdaten_stromverbrauch(params = list(page = 1, pageSize = 5))

## End(Not run)
```

mudab_parameters *List MUDAB parameters*

Description

List MUDAB parameters

Usage

```
mudab_parameters(
  filter = NULL,
  range = NULL,
  orderby = NULL,
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

filter	Optional filter definition.
range	Optional range specification.
orderby	Optional ordering specification.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns measurement parameters from the MUDAB database. Official docs: https://mudab.api.bund.dev.

Value

A tibble with parameters.

Examples

```
## Not run:
mudab_parameters(range = list(from = 0, count = 5))

## End(Not run)
```

mudab_parameters_biologie *List MUDAB parameters (Biologie)*

Description

List MUDAB parameters (Biologie)

Usage

```
mudab_parameters_biologie(
  filter = NULL,
  range = NULL,
  orderby = NULL,
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

filter	Optional filter definition.
range	Optional range specification.
orderby	Optional ordering specification.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns parameter entries for the Biologie compartment. Official docs: https://mudab.api.bund.dev.

Value

A tibble with parameters.

Examples

```
## Not run:
mudab_parameters_biologie(range = list(from = 0, count = 5))

## End(Not run)
```

mudab_parameters_biota *List MUDAB parameters (Biota)*

Description

List MUDAB parameters (Biota)

Usage

```
mudab_parameters_biota(
  filter = NULL,
  range = NULL,
  orderby = NULL,
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

filter	Optional filter definition.
range	Optional range specification.
orderby	Optional ordering specification.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns parameter entries for the Biota compartment. Official docs: https://mudab.api.bund.dev.

Value

A tibble with parameters.

Examples

```
## Not run:
mudab_parameters_biota(range = list(from = 0, count = 5))

## End(Not run)
```

mudab_parameters_sediment *List MUDAB parameters (Sediment)*

Description

List MUDAB parameters (Sediment)

Usage

```
mudab_parameters_sediment(
  filter = NULL,
  range = NULL,
  orderby = NULL,
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

filter	Optional filter definition.
range	Optional range specification.
orderby	Optional ordering specification.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns parameter entries for the Sediment compartment. Official docs: https://mudab.api.bund.dev.

Value

A tibble with parameters.

Examples

```
## Not run:
mudab_parameters_sediment(range = list(from = 0, count = 5))

## End(Not run)
```

mudab_parameters_wasser *List MUDAB parameters (Wasser)*

Description

List MUDAB parameters (Wasser)

Usage

```
mudab_parameters_wasser(
  filter = NULL,
  range = NULL,
  orderby = NULL,
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

filter	Optional filter definition.
range	Optional range specification.
orderby	Optional ordering specification.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns parameter entries for the Wasser compartment. Official docs: https://mudab.api.bund.dev.

Value

A tibble with parameters.

Examples

```
## Not run:
mudab_parameters_wasser(range = list(from = 0, count = 5))

## End(Not run)
```

mudab_parameter_values *List MUDAB parameter values*

Description

List MUDAB parameter values

Usage

```
mudab_parameter_values(
  filter = NULL,
  range = NULL,
  orderby = NULL,
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

filter	Optional filter definition.
range	Optional range specification.
orderby	Optional ordering specification.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns measurement values for parameters from the MUDAB database. Official docs: https://mudab.api.bund.dev.

Value

A tibble with parameter values.

Includes datetime_time as POSIXct in Europe/Berlin when date/time fields are present.

Examples

```
## Not run:
mudab_parameter_values(range = list(from = 0, count = 5))

## End(Not run)
```

mudab_plc_measurements *List MUDAB PLC measurement values*

Description

List MUDAB PLC measurement values

Usage

```
mudab_plc_measurements(
  filter = NULL,
  range = NULL,
  orderby = NULL,
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

filter	Optional filter definition.
range	Optional range specification.
orderby	Optional ordering specification.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns PLC station measurement values. Official docs: https://mudab.api.bund.dev.

Value

A tibble with PLC measurements.

Examples

```
## Not run:
mudab_plc_measurements(range = list(from = 0, count = 5))

## End(Not run)
```

`mudab_plc_parameters` *List MUDAB PLC parameters*

Description

List MUDAB PLC parameters

Usage

```
mudab_plc_parameters(
  filter = NULL,
  range = NULL,
  orderby = NULL,
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

`filter`	Optional filter definition.
`range`	Optional range specification.
`orderby`	Optional ordering specification.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns parameters measured at PLC stations. Official docs: https://mudab.api.bund.dev.

Value

A tibble with PLC parameters.

Examples

```
## Not run:
mudab_plc_parameters(range = list(from = 0, count = 5))

## End(Not run)
```

`mudab_plc_stations` *List MUDAB PLC stations*

Description

List MUDAB PLC stations

Usage

```
mudab_plc_stations(
  filter = NULL,
  range = NULL,
  orderby = NULL,
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

`filter`	Optional filter definition.
`range`	Optional range specification.
`orderby`	Optional ordering specification.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns HELCOM PLC stations. Official docs: https://mudab.api.bund.dev.

Value

A tibble with PLC stations.

Examples

```
## Not run:
mudab_plc_stations(range = list(from = 0, count = 5))

## End(Not run)
```

`mudab_project_stations` *List MUDAB project stations*

Description

List MUDAB project stations

Usage

```
mudab_project_stations(
  filter = NULL,
  range = NULL,
  orderby = NULL,
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`filter`	Optional filter definition.
`range`	Optional range specification.
`orderby`	Optional ordering specification.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns project stations from the MUDAB database. Official docs: https://mudab.api.bund.dev.

Value

A tibble with project stations.

Examples

```
## Not run:
mudab_project_stations(range = list(from = 0, count = 5))

## End(Not run)
```

`mudab_stations` *List MUDAB stations*

Description

List MUDAB stations

Usage

```
mudab_stations(
  filter = NULL,
  range = NULL,
  orderby = NULL,
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

`filter`	Optional filter definition.
`range`	Optional range specification.
`orderby`	Optional ordering specification.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns measurement stations from the MUDAB database. Official docs: https://mudab.api.bund.dev.

Value

A tibble with stations.

Examples

```
## Not run:
mudab_stations(range = list(from = 0, count = 5))

## End(Not run)
```

nina_archive_mowas *Get MOWAS archive entry*

Description

Get MOWAS archive entry

Usage

```
nina_archive_mowas(identifier, safe = TRUE, refresh = FALSE)
```

Arguments

identifier	Warning identifier.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Value

A tibble with archive entry data.

nina_archive_mowas_mapping *Get MOWAS archive mapping*

Description

Get MOWAS archive mapping

Usage

```
nina_archive_mowas_mapping(identifier, safe = TRUE, refresh = FALSE)
```

Arguments

identifier	Warning identifier.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Value

A tibble with archive mapping data.

nina_covid_infos *Get COVID info data*

Description

Get COVID info data

Usage

```
nina_covid_infos(safe = TRUE, refresh = FALSE)
```

Arguments

safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Value

A tibble with COVID info data.

nina_covid_map *Get COVID map data*

Description

Get COVID map data

Usage

```
nina_covid_map(safe = TRUE, refresh = FALSE)
```

Arguments

safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Value

A tibble with COVID map data.

nina_covid_rules *Get COVID rules for an ARS*

Description

Get COVID rules for an ARS

Usage

```
nina_covid_rules(ars, safe = TRUE, refresh = FALSE)
```

Arguments

ars	ARS code.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Value

A tibble with COVID rules data.

nina_covid_ticker *Get COVID ticker*

Description

Get COVID ticker

Usage

```
nina_covid_ticker(safe = TRUE, refresh = FALSE)
```

Arguments

safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Value

A tibble with COVID ticker data.

nina_covid_ticker_message *Get COVID ticker message*

Description

Get COVID ticker message

Usage

```
nina_covid_ticker_message(id, safe = TRUE, refresh = FALSE)
```

Arguments

id	Ticker message id.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Value

A tibble with ticker message data.

nina_dashboard *Get NINA dashboard data*

Description

Get NINA dashboard data

Usage

```
nina_dashboard(ars, safe = TRUE, refresh = FALSE)
```

Arguments

ars	ARS code.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns dashboard data for the given ARS code. Official docs: https://nina.api.bund.dev.

Value

A tibble with dashboard payload.

nina_event_code *Get an event code file*

Description

Get an event code file

Usage

```
nina_event_code(filename, safe = TRUE, refresh = FALSE)
```

Arguments

filename	Event code filename.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Value

A tibble with raw event code bytes.

nina_event_codes *List NINA event codes*

Description

List NINA event codes

Usage

```
nina_event_codes(safe = TRUE, refresh = FALSE)
```

Arguments

safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Value

A tibble with event codes.

Includes last_modification_time as POSIXct in Europe/Berlin.

nina_faqs *List FAQs*

Description

List FAQs

Usage

```
nina_faqs(safe = TRUE, refresh = FALSE)
```

Arguments

safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Value

A tibble with FAQs.

Includes last_modification_time as POSIXct in Europe/Berlin.

nina_logo *Get a logo file*

Description

Get a logo file

Usage

```
nina_logo(filename, safe = TRUE, refresh = FALSE)
```

Arguments

filename	Logo file name.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Value

A tibble with raw logo bytes.

nina_logos *List NINA logos*

Description

List NINA logos

Usage

```
nina_logos(safe = TRUE, refresh = FALSE)
```

Arguments

safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Value

A tibble with logo metadata.

Includes last_modification_time as POSIXct in Europe/Berlin.

nina_mapdata *List map data*

Description

List map data

Usage

```
nina_mapdata(
  source = c("katwarn", "biwapp", "mowas", "dwd", "lhp", "police"),
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

source	Map data source.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Valid sources are: katwarn, biwapp, mowas, dwd, lhp, police.

Value

A tibble with map data entries.

Includes `start_date_time` as POSIXct in Europe/Berlin.

`nina_mowas_rss` *Get MOWAS RSS feed*

Description

Get MOWAS RSS feed

Usage

```
nina_mowas_rss(ars, safe = TRUE, refresh = FALSE)
```

Arguments

`ars`	ARS code.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Value

A tibble with RSS XML text.

`nina_notfalltipps` *List emergency tips*

Description

List emergency tips

Usage

```
nina_notfalltipps(safe = TRUE, refresh = FALSE)
```

Arguments

`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Value

A tibble with emergency tips.

Includes `last_modification_time` as POSIXct in Europe/Berlin.

nina_version *Get data version info*

Description

Get data version info

Usage

```
nina_version(safe = TRUE, refresh = FALSE)
```

Arguments

safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Value

A tibble with version data.

Includes last_modification_time as POSIXct in Europe/Berlin.

nina_warning *Get a NINA travel warning by content id*

Description

Get a NINA travel warning by content id

Usage

```
nina_warning(content_id, safe = TRUE, refresh = FALSE)
```

Arguments

content_id	Travel warning content id.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns full travel warning content, including HTML blocks.

Value

A tibble with warning details.

Includes last_modified_time and effective_time as POSIXct in Europe/Berlin.

See Also

nina_warnings() for ids.

Examples

```
## Not run:
warnings <- nina_warnings()
nina_warning(warnings$content_id[[1]])

## End(Not run)
```

---

nina_warnings *List NINA travel warnings*

Description

List NINA travel warnings

Usage

```
nina_warnings(safe = TRUE, refresh = FALSE)
```

Arguments

safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

The NINA API provides warnings and app data from the Bundesamt fuer Bevoelkerungsschutz. This endpoint returns a list of travel warnings with metadata. Official docs: https://nina.api.bund.dev.

Value

A tibble with warning metadata.

Includes last_modified_time and effective_time as POSIXct in Europe/Berlin.

See Also

nina_warning() for a single warning and nina_mapdata() for map-based alerts.

Examples

```
## Not run:
nina_warnings()

## End(Not run)
```

nina_warning_geojson *Get a NINA warning (GeoJSON)*

Description

Get a NINA warning (GeoJSON)

Usage

```
nina_warning_geojson(identifier, safe = TRUE, refresh = FALSE)
```

Arguments

identifier	Warning identifier.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns a warning in GeoJSON format.

Value

A tibble with geojson payload.

nina_warning_json *Get a NINA warning (JSON)*

Description

Get a NINA warning (JSON)

Usage

```
nina_warning_json(identifier, safe = TRUE, refresh = FALSE)
```

Arguments

identifier	Warning identifier.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns a warning in JSON format.

Value

A tibble with warning details.

Includes `sent_time` as POSIXct in Europe/Berlin.

See Also

`nina_mapdata()` to discover identifiers.

Examples

```
## Not run:
map <- nina_mapdata("mowas")
nina_warning_json(map$id[[1]])

## End(Not run)
```

---

pegel_online_measurements

Get Pegel-Online measurements

Description

Get Pegel-Online measurements

Usage

```
pegel_online_measurements(
  station,
  timeseries,
  start = NULL,
  end = NULL,
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

`station`	Station UUID, name, or number.
`timeseries`	Timeseries shortname.
`start`	Start timestamp in ISO 8601.
`end`	End timestamp in ISO 8601.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns measurement values for a station timeseries. Timestamps must be ISO 8601 strings. Official docs: https://pegel-online.api.bund.dev.

Value

A tibble with measurements.

Includes `timestamp_time` as POSIXct in Europe/Berlin.

See Also

`pegel_online_timeseries()` for metadata and `pegel_online_measurements_plot()` for PNG plots.

Examples

```
## Not run:
stations <- pegel_online_stations(params = list(limit = 1))
pegel_online_measurements(stations$uuid[[1]], "W",
  start = "2024-01-01T00:00:00Z",
  end = "2024-01-02T00:00:00Z"
)

## End(Not run)
```

---

pegel_online_measurements_plot

Get Pegel-Online measurements plot

Description

Get Pegel-Online measurements plot

Usage

```
pegel_online_measurements_plot(
  station,
  timeseries,
  start = NULL,
  end = NULL,
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

station	Station UUID, name, or number.
timeseries	Timeseries shortname.
start	Start timestamp in ISO 8601.
end	End timestamp in ISO 8601.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns a PNG plot for the measurements endpoint. Official docs: https://pegel-online.api.bund.dev.

Value

A tibble with raw PNG bytes.

See Also

pegel_online_measurements() for numeric values.

Examples

```
## Not run:
stations <- pegel_online_stations(params = list(limit = 1))
pegel_online_measurements_plot(stations$uuid[[1]], "W",
  start = "2024-01-01T00:00:00Z",
  end = "2024-01-02T00:00:00Z"
)

## End(Not run)
```

---

pegel_online_station *Get a Pegel-Online station*

Description

Get a Pegel-Online station

Usage

```
pegel_online_station(
  station,
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

station	Station UUID, name, or number.
params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Fetches a single station record. Official docs: https://pegel-online.api.bund.dev.

Value

A tibble with station metadata.

See Also

pegel_online_stations() and pegel_online_timeseries().

Examples

```
## Not run:
stations <- pegel_online_stations(params = list(limit = 1))
pegel_online_station(stations$uuid[[1]])

## End(Not run)
```

---

pegel_online_stations *List Pegel-Online stations*

Description

List Pegel-Online stations

Usage

```
pegel_online_stations(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

The Pegel-Online API provides water level station metadata. Use query parameters to filter stations by water, ids, or location. Official docs: https://pegel-online.api.bund.dev.

Value

A tibble with station metadata.

See Also

pegel_online_station() for a single station and pegel_online_measurements() for time series values.

Examples

```
## Not run:
pegel_online_stations(params = list(limit = 5))

## End(Not run)
```

---

pegel_online_timeseries

Get Pegel-Online timeseries metadata

Description

Get Pegel-Online timeseries metadata

Usage

```
pegel_online_timeseries(
  station,
  timeseries,
  params = list(),
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

station	Station UUID, name, or number.
timeseries	Timeseries shortname.
params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns metadata for a timeseries, including unit and gauge zero. Official docs: https://pegelonline.api.bund.dev.

Value

A tibble with timeseries metadata.

See Also

pegel_online_measurements() for measurement values.

Examples

```
## Not run:
stations <- pegel_online_stations(params = list(limit = 1))
pegel_online_timeseries(stations$uuid[[1]], "W")

## End(Not run)
```

---

pegel_online_waters *List Pegel-Online waters*

Description

List Pegel-Online waters

Usage

```
pegel_online_waters(
  params = list(),
  safe = TRUE,
  refresh = FALSE,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

params	Query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.
flatten	Logical; drop nested list columns.
flatten_mode	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Lists waters available in Pegel-Online. Official docs: https://pegel-online.api.bund.dev.

Value

A tibble with water metadata.

See Also

pegel_online_stations() for station metadata.

Examples

```
## Not run:
pegel_online_waters(params = list(limit = 5))

## End(Not run)
```

---

psm_anwendungen *List approved applications*

Description

List approved applications

Usage

```
psm_anwendungen(
  kennr = NULL,
  awg_id = NULL,
  params = list(),
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

kennr	Optional product identification number.
awg_id	Optional application identifier (16 characters).
params	Additional query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns approved applications (Anwendungsgebiete) which define the combinations of products, crops, and pests for which use is permitted.

Value

A tibble with application data.

See Also

psm_mittel() to list products.

Examples

```
## Not run:
psm_anwendungen(kennr = "024780-00")

## End(Not run)
```

---

psm_kultur_gruppen *List crop groups*

Description

List crop groups

Usage

```
psm_kultur_gruppen(params = list(), safe = TRUE, refresh = FALSE)
```

Arguments

params	Additional query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns hierarchical crop group classifications.

Value

A tibble with crop group data.

Examples

```
## Not run:
psm_kultur_gruppen()

## End(Not run)
```

`psm_mittel` *List approved plant protection products*

Description

List approved plant protection products

Usage

```
psm_mittel(kennr = NULL, params = list(), safe = TRUE, refresh = FALSE)
```

Arguments

`kennr`	Optional product identification number (9 characters).
`params`	Additional query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

The Pflanzenschutzmittelzulassung API provides access to Germany's plant protection product database from the Bundesamt fuer Verbraucherschutz und Lebensmittelsicherheit (BVL). This function returns approved pesticides. Official docs: https://github.com/bundesAPI/pflanzenschutzmittelzulassung-api.

Value

A tibble with plant protection product data.

See Also

`psm_wirkstoffe()` to list active ingredients, `psm_stand()` for data version.

Examples

```
## Not run:
psm_mittel()
psm_mittel(kennr = "024780-00")

## End(Not run)
```

`psm_schadorg_gruppen` *List pest groups*

Description

List pest groups

Usage

```
psm_schadorg_gruppen(params = list(), safe = TRUE, refresh = FALSE)
```

Arguments

`params`	Additional query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns hierarchical pest/pathogen group classifications.

Value

A tibble with pest group data.

Examples

```
## Not run:
psm_schadorg_gruppen()

## End(Not run)
```

psm_stand *Get data version*

Description

Get data version

Usage

```
psm_stand(safe = TRUE, refresh = FALSE)
```

Arguments

safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns the release date/version of the plant protection product database.

Value

A tibble with version information.

Examples

```
## Not run:
psm_stand()

## End(Not run)
```

psm_wirkstoffe *List active ingredients*

Description

List active ingredients

Usage

```
psm_wirkstoffe(
  wirkstoffId = NULL,
  params = list(),
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

wirkstoffId	Optional active ingredient ID.
params	Additional query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns active ingredients (Wirkstoffe) from the plant protection product database.

Value

A tibble with active ingredient data.

See Also

psm_mittel() to list products.

Examples

```
## Not run:
psm_wirkstoffe()

## End(Not run)
```

---

regionalatlas_query *Query Regionalatlas data*

Description

Query Regionalatlas data

Usage

```
regionalatlas_query(
  table,
  where = "1=1",
  out_fields = "*",
  return_geometry = FALSE,
  params = list(),
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

table	Table name containing indicators (e.g., "ai002_1_5" for population density). See Details for common tables.
where	Optional SQL WHERE clause for filtering.
out_fields	Fields to return (default "*" for all).
return_geometry	Logical; include geometry in response.
params	Additional query parameters.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

The Regionalatlas API provides access to over 160 regional indicators from the German statistical offices. Data is available at various administrative levels (Bundeslaender, Regierungsbezirke, Kreise, Gemeinden). Official docs: https://github.com/bundesAPI/regionalatlas-api.

Common tables and their indicators:

- ai002_1_5: Population (ai0201=density, ai0202=change, ai0208=foreigners %)
- ai002_4_5: Age (ai0218=average age, ai0219=mother age at 1st child)
- ai008_1_5: Employment (ai0801=unemployment rate)
- ai_s_01: Disposable income per capita
- ai_s_04: SGB-II quota
- ai017_1: GDP per employee
- ai005: Federal election results

Regional levels in data: typ 1=Bundeslaender, 2=Regierungsbezirke, 3=Kreise, 5=Gemeinden. Filter using WHERE clause, e.g., where = "typ = 1".

Value

A tibble with regional indicator data.

See Also

bunddev_parameters() for available query parameters.

Examples

```
## Not run:
# Population density indicators
regionalatlas_query("ai002_1_5")

# Filter for Bundeslaender only
regionalatlas_query("ai002_1_5", where = "typ = 1")
```

```
# Age data for Kreise
regionalatlas_query("ai002_4_5", where = "typ = 3")

## End(Not run)
```

`smard_indices` *List available SMARD timestamps*

Description

List available SMARD timestamps

Usage

```
smard_indices(
  filter,
  region = "DE",
  resolution = "hour",
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

`filter`	Filter id.
`region`	Region code.
`resolution`	Data resolution.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

The SMARD API provides power market data published by the Bundesnetzagentur. Use this helper to retrieve available timestamps for a given filter/region and resolution. Official docs: https://smard.api.bund.dev.

Note: The registry rate limit states that more than 60 requests per hour are not permitted. Use `safe = TRUE` to respect the limit.

Value

A tibble of timestamps.

See Also

`smard_timeseries()` and `smard_table()` for data retrieval, and `bunddev_parameters()` for parameter discovery.

Examples

```
## Not run:
smard_indices(410, region = "DE", resolution = "hour")

## End(Not run)
```

`smard_table` *Fetch SMARD table data*

Description

Fetch SMARD table data

Usage

```
smard_table(filter, region = "DE", timestamp, safe = TRUE, refresh = FALSE)
```

Arguments

`filter`	Filter id.
`region`	Region code.
`timestamp`	Timestamp from indices (ms), POSIXct, or Date. Timestamps are interpreted in the Europe/Berlin timezone.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns table-style SMARD data for a single timestamp. Use `smard_indices()` to obtain a valid timestamp. Official docs: https://smard.api.bund.dev.

Value

A tibble of table values.

Includes a `time` column with POSIXct timestamps in Europe/Berlin.

See Also

`smard_indices()` for timestamps and `smard_timeseries()` for time series.

Examples

```
## Not run:
indices <- smard_indices(410, region = "DE", resolution = "hour")
smard_table(410, region = "DE", timestamp = indices$timestamp[[1]])

## End(Not run)
```

`smard_timeseries` *Fetch SMARD timeseries data*

Description

Fetch SMARD timeseries data

Usage

```
smard_timeseries(
  filter,
  region = "DE",
  resolution = "hour",
  timestamp,
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

`filter`	Filter id.
`region`	Region code.
`resolution`	Data resolution.
`timestamp`	Timestamp from indices (ms), POSIXct, or Date. Timestamps are interpreted in the Europe/Berlin timezone.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns a tidy time series for a single filter/region. Use `smard_indices()` to obtain a valid timestamp. Official docs: https://smard.api.bund.dev.

Value

A tibble of time series values.

Includes a `time` column with POSIXct timestamps in Europe/Berlin.

See Also

`smard_indices()` for timestamps and `smard_table()` for table output.

Examples

```
## Not run:
indices <- smard_indices(410, region = "DE", resolution = "hour")
smard_timeseries(410, region = "DE", resolution = "hour", timestamp = indices$timestamp[[1]])

## End(Not run)
```

`tagesschau_channels` *Fetch Tagesschau channels*

Description

Fetch Tagesschau channels

Usage

```
tagesschau_channels(flatten = FALSE, flatten_mode = "json")
```

Arguments

`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Lists the Tagesschau channels endpoint. Official docs: https://bundesapi.github.io/tagesschau-api/.

Value

A tibble with channels.

Includes `date_time` as POSIXct in Europe/Berlin.

See Also

`tagesschau_news()` and `tagesschau_homepage()`.

Examples

```
## Not run:
tagesschau_channels(flatten = TRUE)

## End(Not run)
```

`tagesschau_homepage` *Fetch Tagesschau homepage items*

Description

Fetch Tagesschau homepage items

Usage

```
tagesschau_homepage(flatten = FALSE, flatten_mode = "json")
```

Arguments

`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Fetches the Tagesschau homepage feed as provided by the ARD Tagesschau API. Official docs: https://bundesapi.github.io/tagesschau-api/.

Note: The registry rate limit allows up to 60 requests per hour. Usage of content is restricted to private, non-commercial use unless otherwise stated by the source (see Tagesschau CC license notes).

Value

A tibble with homepage items.

Includes `date_time` as POSIXct in Europe/Berlin.

See Also

`tagesschau_news()`, `tagesschau_search()`, and `tagesschau_channels()`.

Examples

```
## Not run:
tagesschau_homepage(flatten = TRUE)

## End(Not run)
```

`tagesschau_news` *Fetch Tagesschau news items*

Description

Fetch Tagesschau news items

Usage

```
tagesschau_news(
  regions = NULL,
  ressort = NULL,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`regions`	Optional region ids.
`ressort`	Optional ressort filter.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Returns current news items with optional filters for region or ressort. Official docs: https://bundesapi.github.io/tagesschau-api/.

Value

A tibble with news items.

Includes `date_time` as POSIXct in Europe/Berlin.

See Also

`tagesschau_homepage()` and `tagesschau_search()`.

Examples

```
## Not run:
tagesschau_news(ressort = "inland", flatten = TRUE)

## End(Not run)
```

`tagesschau_search` *Search Tagesschau content*

Description

Search Tagesschau content

Usage

```
tagesschau_search(
  search_text = NULL,
  page_size = NULL,
  result_page = NULL,
  flatten = FALSE,
  flatten_mode = "json"
)
```

Arguments

`search_text`	Query string.
`page_size`	Results per page.
`result_page`	Result page index.
`flatten`	Logical; drop nested list columns.
`flatten_mode`	Flatten strategy for list columns. Use "unnest" to expand list-columns into multiple rows.

Details

Searches Tagesschau content by free-text query. Official docs: https://bundesapi.github.io/tagesschau-api/.

Value

A tibble with search results.

Includes `date_time` as POSIXct in Europe/Berlin.

See Also

`tagesschau_news()` and `tagesschau_homepage()`.

Examples

```
## Not run:
tagesschau_search(search_text = "energie", page_size = 10, flatten = TRUE)

## End(Not run)
```

travelwarning_healthcare *List healthcare documents*

Description

List healthcare documents

Usage

```
travelwarning_healthcare(safe = TRUE, refresh = FALSE)
```

Arguments

safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns healthcare documents provided by the Auswaertiges Amt. Official docs: https://travelwarning.api.bund.dev.

Value

A tibble with healthcare entries.

Includes `last_modified_time` as POSIXct in Europe/Berlin.

Examples

```
## Not run:
travelwarning_healthcare()

## End(Not run)
```

travelwarning_representatives_country *List German representatives in foreign countries*

Description

List German representatives in foreign countries

Usage

```
travelwarning_representatives_country(safe = TRUE, refresh = FALSE)
```

Arguments

safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns a list of German representatives in foreign countries. Official docs: https://travelwarning.api.bund.dev.

Value

A tibble with representatives.

Includes last_modified_time as POSIXct in Europe/Berlin.

Examples

```
## Not run:
travelwarning_representatives_country()

## End(Not run)
```

---

travelwarning_representatives_germany

List foreign representatives in Germany

Description

List foreign representatives in Germany

Usage

```
travelwarning_representatives_germany(safe = TRUE, refresh = FALSE)
```

Arguments

safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns a list of foreign representatives in Germany. Official docs: https://travelwarning.api.bund.dev.

Value

A tibble with representatives.

Includes last_modified_time as POSIXct in Europe/Berlin.

Examples

```
## Not run:
travelwarning_representatives_germany()

## End(Not run)
```

`travelwarning_state_names` *List state names documents*

Description

List state names documents

Usage

```
travelwarning_state_names(safe = TRUE, refresh = FALSE)
```

Arguments

`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns documents with state name information. Official docs: https://travelwarning.api.bund.dev.

Value

A tibble with state name entries.

Includes `last_modified_time` as POSIXct in Europe/Berlin.

Examples

```
## Not run:
travelwarning_state_names()

## End(Not run)
```

`travelwarning_warning` *Get a travel warning by content id*

Description

Get a travel warning by content id

Usage

```
travelwarning_warning(content_id, safe = TRUE, refresh = FALSE)
```

Arguments

`content_id`	Travel warning content id.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns the full travel warning content, including HTML blocks. Official docs: https://travelwarning.api.bund.dev.

Value

A tibble with travel warning details.

Includes `last_modified_time` and `effective_time` as POSIXct in Europe/Berlin.

See Also

`travelwarning_warnings()` to list available ids.

Examples

```
## Not run:
warnings <- travelwarning_warnings()
travelwarning_warning(warnings$content_id[[1]])

## End(Not run)
```

`travelwarning_warnings` *List travel warnings*

Description

List travel warnings

Usage

```
travelwarning_warnings(safe = TRUE, refresh = FALSE)
```

Arguments

`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

The travelwarning API provides travel and safety information from the Auswaertiges Amt. This endpoint returns a list of all travel warnings with metadata. Official docs: https://travelwarning.api.bund.dev.

Value

A tibble with travel warnings.

Includes `last_modified_time` and `effective_time` as POSIXct in Europe/Berlin.

See Also

`travelwarning_warning()` for full details of a single warning.

Examples

```
## Not run:
travelwarning_warnings()

## End(Not run)
```

`weiterbildungssuche_facetten` *Get faceted overview of courses*

Description

Get faceted overview of courses

Usage

```
weiterbildungssuche_facetten(
  sw = NULL,
  params = list(),
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

`sw`	Search keyword.
`params`	Additional query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns a faceted overview of available training offerings with counts by category.

Value

A tibble with facet data.

See Also

`weiterbildungssuche_search()` for full course search.

Examples

```
## Not run:
weiterbildungssuche_facetten(sw = "BWL")

## End(Not run)
```

`weiterbildungssuche_search` *Search continuing education courses*

Description

Search continuing education courses

Usage

```
weiterbildungssuche_search(
  sw = NULL,
  orte = NULL,
  uk = NULL,
  page = NULL,
  params = list(),
  safe = TRUE,
  refresh = FALSE
)
```

Arguments

`sw`	Search keyword.
`orte`	Location with coordinates (format: "City_lat_lon").
`uk`	Search radius in km (25, 50, 100, 150, 200, or "bundesweit").
`page`	Result page number.
`params`	Additional query parameters.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

The Weiterbildungssuche API provides access to one of Germany's largest continuing education databases from the Bundesagentur fuer Arbeit. Authentication uses the public client id `infosysbub-wbsuche` as X-API-Key. Official docs: https://bundesapi.github.io/weiterbildungssuche-api/.

Common parameters include:

- `sys`: C = vocational, FW = advanced, empty = both
- `uz`: 1 = full-time, 2 = part-time
- `uf`: 1 = in-person, 2 = online, 3 = blended, 4 = distance
- `bg`: 1 = education voucher eligible

Value

A tibble with course offerings.

See Also

weiterbildungssuche_facetten() for faceted search overview.

Examples

```
## Not run:
weiterbildungssuche_search(sw = "Projektmanagement")
weiterbildungssuche_search(sw = "IT", orte = "Berlin_52.52_13.405", uk = 50)

## End(Not run)
```

---

zoll_kategorien *Get product categories*

Description

Get product categories

Usage

```
zoll_kategorien(last_modified_date = NULL, safe = TRUE, refresh = FALSE)
```

Arguments

last_modified_date	Optional datetime to retrieve only changes after this date.
safe	Logical; apply throttling and caching.
refresh	Logical; refresh cached responses.

Details

Returns product category database.

Value

A tibble with category data.

Examples

```
## Not run:
zoll_kategorien()

## End(Not run)
```

`zoll_kurse` *Get exchange rates for customs calculations*

Description

Get exchange rates for customs calculations

Usage

```
zoll_kurse(last_modified_date = NULL, safe = TRUE, refresh = FALSE)
```

Arguments

`last_modified_date`	Optional datetime to retrieve only changes after this date (format: "YYYY-MM-DD" or "YYYY-MM-DDTHH:MM:SS").
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

The Zoll API provides import tariff and exchange rate data from the German Federal Ministry of Finance. This function returns currency exchange rates used for customs calculations. Official docs: https://zoll.api.bund.dev.

Value

A tibble with exchange rate data.

See Also

`zoll_produkte()` for product tariffs, `zoll_laender()` for country data.

Examples

```
## Not run:
zoll_kurse()

## End(Not run)
```

`zoll_laender` *Get country data for customs*

Description

Get country data for customs

Usage

```
zoll_laender(last_modified_date = NULL, safe = TRUE, refresh = FALSE)
```

Arguments

`last_modified_date`	Optional datetime to retrieve only changes after this date.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns country information and classifications for customs purposes.

Value

A tibble with country data.

See Also

`zoll_produkte()` for products.

Examples

```
## Not run:
zoll_laender()

## End(Not run)
```

`zoll_produkte` *Get product tariff data*

Description

Get product tariff data

Usage

```
zoll_produkte(last_modified_date = NULL, safe = TRUE, refresh = FALSE)
```

Arguments

`last_modified_date`	Optional datetime to retrieve only changes after this date.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns product database with import tariff rates.

Value

A tibble with product tariff data.

See Also

`zoll_kurse()` for exchange rates, `zoll_produktgruppen()` for product groups.

Examples

```
## Not run:
zoll_produkte()

## End(Not run)
```

`zoll_produktgruppen` *Get product groups*

Description

Get product groups

Usage

```
zoll_produktgruppen(last_modified_date = NULL, safe = TRUE, refresh = FALSE)
```

Arguments

`last_modified_date`	Optional datetime to retrieve only changes after this date.
`safe`	Logical; apply throttling and caching.
`refresh`	Logical; refresh cached responses.

Details

Returns product group classifications.

Value

A tibble with product group data.

Examples

```
## Not run:
zoll_produktgruppen()

## End(Not run)
```

Index