Package: tidybrreg (via r-universe)

June 6, 2026

Title Tidy Access to Norway's Business Registry ('Enhetsregisteret')

Version 0.3.8

Description A tidy interface to download, search, and analyze data from the Norwegian Central Coordinating Register for Legal Entities (Enhetsregisteret), maintained by the Brønnøysund Register Centre. Provides access to registration data for approximately 1 million legal entities including companies, partnerships, sole proprietorships, associations, and government bodies. Returns tibbles with English column names mapped via a data-driven dictionary; unknown API fields pass through with auto-generated names. Coded values (legal forms, industry codes, roles) can be translated to English on demand using bundled reference data or live lookups from the SSB Klass API. Data is freely available under the Norwegian Licence for Open Government Data (NLOD 2.0).

License MIT + file LICENSE

URL https://sondreskarsten.github.io/tidybrreg/, https://github.com/sondreskarsten/tidybrreg

BugReports https://github.com/sondreskarsten/tidybrreg/issues

Depends R (>= 4.1.0)

Imports cli, dplyr, httr2 (>= 1.0.0), jsonlite, readr, rlang, tibble

Suggests arrow (>= 12.0.0), nanoparquet (>= 0.3.0), curl, duckdb, klassR, knitr, rmarkdown, tidygraph, tidyr, tsibble, yyjsonr, httptest2, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/pak/sysreqs libssl-dev libx11-dev

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

Date/Publication 2026-06-06 20:53:57 UTC

RemoteUrl https://github.com/sondreskarsten/tidybrreg

RemoteRef HEAD

RemoteSha 490519fb1a092e53d32c63785b372f3d42db6a28

Contents

`annotation_infotypes` *Annotation infotype codes with English descriptions*

Description

Maps brreg påtegning `infotype` codes to English descriptions. Used by `brreg_annotations()` when `translate = TRUE`. Covers the infotype codes documented in the brreg API reference (`"NAVN"`, `"FADR"`) and those observed in live data (role codes used for missing-role annotations); unknown codes pass through unchanged. Each annotation's own Norwegian text is available in the `tekst` column of `brreg_annotations()` output.

Usage

```
annotation_infotypes
```

Format

A tibble with 7 rows and 2 columns:

code Annotation infotype code (e.g. `"FADR"`, `"NAVN"`).

name_en English description.

See Also

`brreg_annotations()` for the function that uses this table.

Other tidybrreg reference data: `field_dict`, `legal_forms`, `role_groups`, `role_types`

Examples

```
annotation_infotypes
```

`as_brreg_tsibble` *Convert tidybrreg output to tsibble*

Description

Convert the output of `brreg_panel()` or `brreg_series()` to a tsibble for use with the tidyverts ecosystem (fable, feasts, slider). Uses `regular = FALSE` since brreg snapshots are irregularly spaced.

Usage

```
as_brreg_tsibble(x, key = NULL, index = NULL)
```

Arguments

`x`	A tibble from `brreg_panel()` or `brreg_series()`.
`key`	Character vector of key column(s). For panels, typically `"org_nr"`. For series, the grouping variable (e.g. `"legal_form"`). If `NULL`, inferred from the `brreg_panel_meta` attribute.
`index`	Character. Name of the time index column. Default `"period"` for series output, `"snapshot_date"` for panel output.

Value

A tsibble.

See Also

`brreg_panel()`, `brreg_series()`.

Other tidybrreg panel functions: `brreg_change_summary()`, `brreg_changes()`, `brreg_events()`, `brreg_flows()`, `brreg_panel()`, `brreg_replay()`, `brreg_series()`

Examples

```
panel <- brreg_panel(cols = c("employees", "legal_form"))
as_brreg_tsibble(panel)
```

brreg_annotation_summary *Count entities with active annotations*

Description

Quick summary of how many entities currently carry påtegninger, grouped by annotation type.

Usage

```
brreg_annotation_summary()
```

Value

A tibble with `infotype` and `n`.

See Also

Other tidybrreg data management functions: `brreg_annotations()`, `brreg_status()`, `brreg_sync()`, `brreg_sync_status()`, `diff_roller_state()`, `read_changelog()`

Examples

```
brreg_annotation_summary()
```

brreg_annotations *Retrieve registry annotations (påtegninger) for entities*

Description

Påtegninger are public annotations placed on entities by the register keeper to warn third parties of irregularities — missing board members, undelivered accounts, deceased officers. They are the earliest formal signal that an entity is in regulatory trouble, preceding forced dissolution warnings by weeks to months.

Usage

```
brreg_annotations(
  org_nr = NULL,
  infotype = NULL,
  active_only = TRUE,
  translate = FALSE
)
```

Arguments

org_nr	Optional character vector of organisation numbers. NULL returns all annotations.
infotype	Optional character vector of annotation type codes to filter by (e.g. "FADR", "NAVN").
active_only	Logical. If TRUE (default), return only annotations currently in force. If FALSE, include cleared annotations from the changelog.
translate	Logical. If TRUE, add an infotype_desc column with English descriptions from annotation_infotypes; unknown codes pass through unchanged. Default FALSE.

Details

Requires brreg_sync() to have been run at least once to populate the local påtegninger state.

Value

A tibble with columns: org_nr, position (array index), infotype, tekst (annotation text), innfoert_dato (date introduced). With translate = TRUE, an infotype_desc column is inserted after infotype.

See Also

brreg_sync() to populate annotation data, brreg_changes() to track annotation events over time, annotation_infotypes for the English infotype lookup table.

Other tidybrreg data management functions: brreg_annotation_summary(), brreg_status(), brreg_sync(), brreg_sync_status(), diff_roller_state(), read_changelog()

Examples

```
brreg_sync()
brreg_annotations()
brreg_annotations(infotype = "FADR")
brreg_annotations(translate = TRUE)
```

---

brreg_board_network *Build a director interlock network*

Description

Construct a bipartite graph of entities and persons linked by board/officer roles. Returns a tbl_graph (tidygraph) object suitable for centrality analysis and ggraph visualization.

Usage

```
brreg_board_network(org_nrs = NULL, roles_data = NULL)
```

Arguments

org_nrs	Character vector of organization numbers to include. Roles are fetched via brreg_roles() for each entity.
roles_data	Optional pre-fetched roles tibble (from brreg_roles()). If provided, org_nrs is ignored.

Details

For a full ego network including sub-units, child entities, and legal roles in addition to board roles, use brreg_network() instead.

Value

A tbl_graph with two node types: "entity" (identified by org_nr) and "person" (identified by person_id). Edge attributes include role_code, role_group_code, and org_nr.

See Also

brreg_network() for full entity network graphs, brreg_roles() to fetch role data, brreg_board_summary() for board-level covariates.

Other tidybrreg governance functions: brreg_network(), brreg_survival_data()

Examples

```
net <- brreg_board_network(c("923609016", "984851006"))
net
```

---

brreg_board_summary *Derive board-level summary covariates from role data*

Description

Compute firm-level variables commonly used in corporate governance research: board size, composition counts, and officer indicators.

Usage

```
brreg_board_summary(roles)
```

Arguments

roles	A tibble returned by brreg_roles().

Value

A 1-row tibble with columns: `org_nr`, `board_size`, `n_chair`, `n_deputy_chair`, `n_members`, `n_alternates`, `n_observers`, `n_employee_elected`, `has_ceo`, `has_auditor`, `auditor_org_nr`. Counts exclude resigned and deregistered roles.

See Also

`brreg_roles()` for the underlying role data.

Other tidybrreg entity functions: `brreg_children()`, `brreg_download()`, `brreg_entity()`, `brreg_roles()`, `brreg_roles_legal()`, `brreg_search()`, `brreg_underenheter()`, `brreg_update_fields()`, `brreg_updates()`

Examples

```
brreg_roles("923609016") |> brreg_board_summary()
```

---

brreg_change_summary *Summarize changes by field and type*

Description

Produces a count table of how many changes occurred per field and change type, useful for understanding the volume and distribution of registry mutations.

Usage

```
brreg_change_summary(from = NULL, to = NULL, registry = NULL)
```

Arguments

`from`, `to`	Date range (inclusive).
`registry`	Character vector of streams to include. Default includes all four.

Value

A tibble with `registry`, `change_type`, `field`, `n`.

See Also

`brreg_changes()` for raw changelog rows, `brreg_sync()` to populate the changelog.

Other tidybrreg panel functions: `as_brreg_tsibble()`, `brreg_changes()`, `brreg_events()`, `brreg_flows()`, `brreg_panel()`, `brreg_replay()`, `brreg_series()`

Examples

```
brreg_change_summary(from = Sys.Date() - 7)
```

brreg_changes *Query the change log for field-level mutations*

Description

Returns a filtered view of all recorded changes across the four sync streams: enheter, underenheter, roller, and påtegninger. Every call to `brreg_sync()` appends events to the changelog; this function reads and filters them.

Usage

```
brreg_changes(
  track = NULL,
  registry = NULL,
  change_type = NULL,
  from = NULL,
  to = NULL,
  org_nr = NULL
)
```

Arguments

`track`	Character vector of fields to include (e.g. `c("nace_1", "municipality_code", "employees")`). `NULL` returns all fields.
`registry`	Character vector of streams to include. Default includes all four.
`change_type`	Character vector of event types to include. Options: `"entry"`, `"exit"`, `"change"`, `"annotation_added"`, `"annotation_cleared"`.
`from, to`	Date range (inclusive).
`org_nr`	Optional character vector of organisation numbers.

Value

A tibble with columns: `timestamp`, `org_nr`, `registry`, `change_type`, `field`, `value_from`, `value_to`, `update_id`.

See Also

`brreg_sync()` to populate the changelog, `brreg_flows()` for aggregated entry/exit counts.

Other tidybrreg panel functions: `as_brreg_tsibble()`, `brreg_change_summary()`, `brreg_events()`, `brreg_flows()`, `brreg_panel()`, `brreg_replay()`, `brreg_series()`

Examples

```
brreg_sync()

# All changes this month
brreg_changes(from = Sys.Date() - 30)

# NACE reclassifications only
brreg_changes(track = "nace_1", change_type = "change")

# Entries and exits for a specific entity
brreg_changes(org_nr = "923609016", change_type = c("entry", "exit"))

# All annotation events
brreg_changes(registry = "paategninger")
```

---

brreg_children *Get child entities in the organisational hierarchy*

Description

Query the enheter registry for entities whose `overordnetEnhet` matches the given org number. This traverses the ORGL parent-child hierarchy (e.g. Stortinget to Riksrevisjonen), which is distinct from the enhet-to-underenhet relationship.

Usage

```
brreg_children(org_nr, max_results = 200, type = c("code", "label"))
```

Arguments

`org_nr`	Character. 9-digit organization number of the parent entity (hovedenhet).
`max_results`	Integer. Maximum sub-units to return (default 200).
`type`	One of `"code"` (default) or `"label"`.

Value

A tibble with one row per child entity. Same column schema as `brreg_search()` with `registry = "enheter"`.

See Also

`brreg_underenheter()` for sub-units (BEDR/AAFY), `brreg_entity()` for single lookups.

Other tidybrreg entity functions: `brreg_board_summary()`, `brreg_download()`, `brreg_entity()`, `brreg_roles()`, `brreg_roles_legal()`, `brreg_search()`, `brreg_underenheter()`, `brreg_update_fields()`, `brreg_updates()`

Examples

```
brreg_children("971524960")
```

`brreg_cleanup` *Remove old snapshots from the local store*

Description

Delete snapshot partitions by count or age. At least one of `keep_n` or `max_age_days` must be provided.

Usage

```
brreg_cleanup(
  keep_n = NULL,
  max_age_days = NULL,
  type = c("enheter", "underenheter", "roller")
)
```

Arguments

`keep_n`	Integer. Keep the `keep_n` most recent snapshots and delete the rest. `NULL` to skip this criterion.
`max_age_days`	Integer. Delete snapshots older than this many days. `NULL` to skip this criterion.
`type`	One of `"enheter"` or `"underenheter"`.

Value

A tibble of deleted snapshots (invisibly).

See Also

`brreg_snapshots()` to list available snapshots, `brreg_data_dir()` for storage location.

Other tidybrreg snapshot functions: `brreg_data_dir()`, `brreg_import()`, `brreg_manifest()`, `brreg_open()`, `brreg_snapshot()`, `brreg_snapshots()`

Examples

```
brreg_cleanup(keep_n = 12)
brreg_cleanup(max_age_days = 365)
```

brreg_data_dir *Path to the tidybrreg snapshot store*

Description

Returns (and creates if needed) the directory where tidybrreg stores Parquet snapshots. Location follows R's standard user data directory convention via `tools::R_user_dir("tidybrreg", "data")`. Override with `options(brreg.data_dir = "/custom/path")`.

Usage

```
brreg_data_dir()
```

Value

Character path.

See Also

`brreg_snapshot()` to save snapshots, `brreg_open()` to read them.

Other tidybrreg snapshot functions: `brreg_cleanup()`, `brreg_import()`, `brreg_manifest()`, `brreg_open()`, `brreg_snapshot()`, `brreg_snapshots()`

Examples

```
brreg_data_dir()
```

brreg_download *Download the full Norwegian business register*

Description

Download a complete extract of the Central Coordinating Register for Legal Entities (~1 million entities, ~145 MB gzipped). The bulk endpoint does not support server-side filtering — it always returns all entities. Use `brreg_search()` for filtered queries up to 10,000 results, or download the full register here and filter locally.

Usage

```
brreg_download(
  type = c("enheter", "underenheter", "roller"),
  format = c("csv", "json"),
  refresh = FALSE,
  cache = TRUE,
  type_output = c("tibble", "arrow", "path")
)
```

Arguments

type	One of "enheter" (main entities, default), "underenheter" (sub-entities / establishments), or "roller" (all roles for all entities). Roller data is only available as JSON via /roller/totalbestand (~131 MB).
format	Download format: "csv" (default for enheter/underenheter, semicolon-delimited) or "json" (JSON array). Roller bulk download is always JSON regardless of this parameter.
refresh	FALSE (default): use cached file if available. TRUE: force re-download. "auto": check ETag and re-download only if server has a newer version.
cache	Logical. If TRUE (default), cache downloaded file persistently.
type_output	One of "tibble" (default), "arrow" (requires the arrow package), or "path" (returns the file path without parsing).

Value

Depends on type_output:

- "tibble": A tibble with ~1 million rows. Column names mapped via field_dict.
- "arrow": An Arrow Table (lazy, not loaded into memory).
- "path": Character file path to the cached CSV.

Backend routing

The brreg API has two data access paths with fundamentally different characteristics, following the cansim (Statistics Canada) pattern of separate functions for separate access patterns rather than the eurostat pattern of auto-routing within one function:

- brreg_search(): JSON API with server-side filtering. Fast for small result sets, capped at 10,000. Interactive exploration.
- brreg_download(): Bulk CSV. Always returns the full register. Appropriate for panel construction, spatial joins, or any analysis requiring more than 10,000 entities.

Results from both paths use the same column names via field_dict.

Caching

Downloaded files are cached in tools::R_user_dir("tidybrreg", "cache") as gzipped CSV. Use refresh = TRUE to force re-download, or refresh = "auto" to re-download only if the cached file is older than the latest nightly bulk export (checked via ETag headers, following the cansim refresh = "auto" pattern).

See Also

brreg_search() for filtered API queries, brreg_updates() for incremental changes since a given date.

Other tidybrreg entity functions: brreg_board_summary(), brreg_children(), brreg_entity(), brreg_roles(), brreg_roles_legal(), brreg_search(), brreg_underenheter(), brreg_update_fields(), brreg_updates()

Examples

```
# Download full register as tibble (~145MB download, ~1M rows)
entities <- brreg_download()

# Just get the file path (no parsing)
path <- brreg_download(type_output = "path")

# Force refresh
entities <- brreg_download(refresh = TRUE)
```

`brreg_entity` *Look up a Norwegian legal entity*

Description

Retrieve registration details for a legal entity from Norway's Central Coordinating Register for Legal Entities (Enhetsregisteret), maintained by the Brønnøysund Register Centre. Every legal entity operating in Norway is assigned a unique 9-digit organization number and registered in this central register.

Usage

```
brreg_entity(
  org_nr,
  registry = c("auto", "enheter", "underenheter"),
  type = c("code", "label")
)
```

Arguments

`org_nr`	Character. A 9-digit Norwegian organization number. Validated using `brreg_validate()` before the API call.
`registry`	One of `"auto"` (default, tries enheter then underenheter), `"enheter"` (main entities only), or `"underenheter"` (sub-entities only). Sub-entities have different fields (e.g. `overordnetEnhet` for the parent entity, `beliggenhetsadresse` instead of `forretningsadresse`).
`type`	A type of variables: `"code"` (default) returns raw codes, `"label"` returns English labels for coded columns (legal form, NACE, sector, etc.), following the eurostat package's pattern.

Details

Column names are translated from Norwegian to English via the package field dictionary (field_dict). API fields not in the dictionary pass through with auto-generated snake_case names, so new fields added by brreg are never silently dropped. Use `brreg_label()` to translate coded values (legal forms, NACE codes) to English descriptions.

Value

A tibble with one row and one column per API field. Column names follow the package field dictionary. Key columns include `org_nr`, `name`, `legal_form`, `employees`, `founding_date`, `nace_1`, `municipality_code`, `bankrupt`, and `parent_org_nr`. For deleted entities (HTTP 410), returns a tibble with columns `org_nr`, `deleted`, and `deletion_date`.

See Also

`brreg_search()` for querying multiple entities, `brreg_label()` for translating codes to English, field_dict for the column name mapping.

Other tidybrreg entity functions: `brreg_board_summary()`, `brreg_children()`, `brreg_download()`, `brreg_roles()`, `brreg_roles_legal()`, `brreg_search()`, `brreg_underenheter()`, `brreg_update_fields()`, `brreg_updates()`

Examples

```
# Equinor ASA — Norway's largest company
brreg_entity("923609016")

# With English labels (eurostat pattern)
brreg_entity("923609016", type = "label")

# Or pipe to brreg_label() for more control
brreg_entity("923609016") |> brreg_label(code = "legal_form")
```

`brreg_events` *Detect changes between two snapshots*

Description

Compare two dated snapshots and return a tibble of entity-level events: entries (new entities), exits (deleted entities), and field-level changes. Unlike the CDC stream, snapshot diffs provide both old and new values for every changed field.

Usage

```
brreg_events(
  date_from,
  date_to,
  cols = NULL,
  type = c("enheter", "underenheter")
)
```

Arguments

`date_from`, `date_to`
: Dates identifying the two snapshots to compare. Both must exist in the snapshot store. `date_from` is the "before" state, `date_to` is the "after" state.

`cols`
: Character vector of columns to track for changes. `NULL` (default) tracks all columns in field_dict.

`type`
: One of `"enheter"` or `"underenheter"`.

Value

A tibble with columns: `org_nr`, `event_type` (`"entry"`, `"exit"`, `"change"`), `event_date` (the `date_to` snapshot date), `field` (column name, `NA` for entry/exit), `value_from` (character, previous value), `value_to` (character, new value).

See Also

`brreg_updates()` for the CDC stream (API-level changes), `brreg_panel()` for full panel construction.

Other tidybrreg panel functions: `as_brreg_tsibble()`, `brreg_change_summary()`, `brreg_changes()`, `brreg_flows()`, `brreg_panel()`, `brreg_replay()`, `brreg_series()`

Examples

```
snaps <- brreg_snapshots()
if (nrow(snaps) >= 2) {
  events <- brreg_events(snaps$snapshot_date[1], snaps$snapshot_date[2])
  events
}
```

---

brreg_flows *Compute daily entry and exit flows*

Description

Calculate daily counts of entity registrations (entries) and deletions (exits) classified by industry (NACE code) and geography (municipality code). Three data paths, selected automatically:

Usage

```
brreg_flows(
  data = NULL,
  updates = NULL,
  by = c("nace_1", "municipality_code"),
  from = NULL,
```

```
  to = NULL,
  legal_form = NULL
)
```

Arguments

data	Optional. A tibble from brreg_download() or a snapshot. Required when no changelog exists. Must contain org_nr, registration_date, nace_1, and municipality_code.
updates	Optional. A tibble from brreg_updates() with CDC events.
by	Character vector of grouping columns. Default c("nace_1", "municipality_code"). Use NULL for national totals, or any column present in data.
from, to	Date range for the output. NULL defaults to the range of observed events.
legal_form	Optional character vector of legal form codes to include (e.g. c("AS", "ENK")). NULL includes all.

Details

1. **Changelog path** (preferred) — when brreg_sync() has been run, reads directly from the persistent changelog. Provides timestamped entries, exits, and field-level transitions. No arguments needed.
2. **Bulk + CDC path** — pass data (from brreg_download()) and optionally updates (from brreg_updates()). Registration dates provide historical entries; CDC provides recent entries + exits.
3. **Bulk-only path** — pass data alone. Only entries are computed (no exit data available).

Value

A tibble with columns: date (Date), grouping columns from by, entries (integer), exits (integer), net (integer: entries - exits). An attribute flow_source records which data sources contributed.

Entry vs. founding date

This function uses registration_date (registreringsdatoEnhetsregisteret), NOT founding_date (stiftelsesdato). Registration date is when the entity entered the registry. Founding date can precede registration by months (AS companies) or years (associations).

See Also

brreg_download() to get bulk data, brreg_updates() to get CDC events, brreg_series() for snapshot-based time series, as_brreg_tsibble() for tsibble conversion.

Other tidybrreg panel functions: as_brreg_tsibble(), brreg_change_summary(), brreg_changes(), brreg_events(), brreg_panel(), brreg_replay(), brreg_series()

Examples

```
entities <- brreg_download()
flows <- brreg_flows(entities)

# With CDC exits
cdc <- brreg_updates(since = "2026-01-01", size = 10000)
flows <- brreg_flows(entities, updates = cdc)

# Monthly by NACE section
flows |>
  dplyr::mutate(month = format(date, "%Y-%m"),
                nace_section = substr(nace_1, 1, 2)) |>
  dplyr::summarise(entries = sum(entries), exits = sum(exits),
                   .by = c(month, nace_section))
```

---

brreg_harmonize_kommune

Harmonize municipality codes across boundary reforms

Description

Remap `municipality_code` (kommunenummer) to the classification valid at a target date, using correspondence tables from SSB's KLASS system (classification 131). Handles the 2020 municipal reform (428 → 356 municipalities) and 2024 county reversals.

Usage

```
brreg_harmonize_kommune(
  data,
  target_date = Sys.Date(),
  col = "municipality_code"
)
```

Arguments

`data`	A tibble with a municipality code column.
`target_date`	Date. Remap all codes to the classification valid at this date. Default: today.
`col`	Column name containing municipality codes. Default `"municipality_code"`.

Value

The input tibble with two added columns: `{col}_harmonized` (the remapped code) and `{col}_target_name` (municipality name at the target date). Unmatched codes pass through unchanged.

See Also

brreg_harmonize_nace() for NACE code harmonization.

Other tidybrreg harmonization functions: brreg_harmonize_nace()

Examples

```
# Remap old codes to current boundaries
df <- tibble::tibble(municipality_code = c("0301", "1201", "0602"))
brreg_harmonize_kommune(df)
```

---

brreg_harmonize_nace *Harmonize NACE industry codes across classification revisions*

Description

Remap NACE codes between SN2007 (NACE Rev. 2) and SN2025 (NACE Rev. 2.1) using SSB KLASS correspondence tables.

Usage

```
brreg_harmonize_nace(data, from = "SN2007", to = "SN2025", col = "nace_1")
```

Arguments

data	A tibble with a NACE code column.
from	Source classification: "SN2007" (default) or "SN2025".
to	Target classification: "SN2025" (default) or "SN2007".
col	Column name containing NACE codes. Default "nace_1".

Value

The input tibble with {col}_harmonized (remapped code) and {col}_ambiguous (logical, TRUE when the mapping is one-to-many and the first match was used). Unmatched codes pass through unchanged.

See Also

brreg_harmonize_kommune() for municipality harmonization.

Other tidybrreg harmonization functions: brreg_harmonize_kommune()

Examples

```
df <- tibble::tibble(nace_1 = c("06.100", "64.190", "62.010"))
brreg_harmonize_nace(df, from = "SN2007", to = "SN2025")
```

`brreg_import` *Import a historical CSV as a snapshot partition*

Description

Read a brreg bulk CSV file (as downloaded by `brreg_download()` or from the brreg website), normalize column names via field_dict, and save as a dated Parquet partition in the snapshot store.

Usage

```
brreg_import(
  path,
  snapshot_date,
  type = c("enheter", "underenheter", "roller"),
  force = FALSE
)
```

Arguments

`path`	Path to a brreg CSV file (gzipped or plain).
`snapshot_date`	The date this CSV represents. Required — the CSV itself contains no date metadata.
`type`	One of `"enheter"` or `"underenheter"`.
`force`	Logical. Overwrite existing partition.

Value

The file path to the written Parquet partition (invisibly).

See Also

`brreg_snapshot()` to download and save today's register.

Other tidybrreg snapshot functions: `brreg_cleanup()`, `brreg_data_dir()`, `brreg_manifest()`, `brreg_open()`, `brreg_snapshot()`, `brreg_snapshots()`

Examples

```
# Import a historical download
brreg_import("enheter_2024-12-31.csv.gz", snapshot_date = "2024-12-31")
```

brreg_label *Translate codes to human-readable English labels*

Description

Replace coded values in a brreg tibble with English descriptions, following the eurostat package's `label_eurostat()` pattern. Works on both data frames and character vectors.

Usage

```
brreg_label(x, dic = NULL, code = NULL, lang = "en")
```

Arguments

x	A tibble from `brreg_entity()`, `brreg_search()`, or `brreg_roles()`, or a character vector of codes.
dic	A character string naming the dictionary to use when x is a character vector. One of `"legal_form"`, `"nace"`, `"sector"`, `"role"`, `"role_group"`. Ignored when x is a data frame (dictionaries are inferred from column names).
code	For data frames: character vector of column names for which to retain the original code alongside the label. A column with suffix `_code` is added. For example, `brreg_label(x, code = "legal_form")` adds `legal_form_code`.
lang	Language for NACE and sector labels. `"en"` (default) or `"no"` (Norwegian original from brreg API).

Value

When x is a data frame: the same tibble with code columns replaced by English labels. When x is a character vector: a character vector of labels.

See Also

legal_forms, role_types, role_groups for bundled reference data, `get_brreg_dic()` for fetching fresh dictionaries.

Other tidybrreg utilities: `brreg_validate()`, `get_brreg_dic()`

Examples

```
eq <- brreg_entity("923609016")

# Label all code columns
brreg_label(eq)

# Keep original codes alongside labels
brreg_label(eq, code = "legal_form")

# Label a character vector directly
```

```
brreg_label(c("AS", "ASA", "ENK"), dic = "legal_form")
```

`brreg_manifest` *Read the snapshot manifest*

Description

Returns the provenance catalog recording every download: endpoint, timestamps, HTTP headers, file hashes, and CDC bridge metadata. The manifest lives at `brreg_data_dir()/manifest.json`.

Usage

```
brreg_manifest()
```

Value

A tibble with one row per snapshot. Returns an empty tibble if no manifest exists.

See Also

`brreg_snapshot()` to create snapshots, `brreg_snapshots()` to list them.

Other tidybrreg snapshot functions: `brreg_cleanup()`, `brreg_data_dir()`, `brreg_import()`, `brreg_open()`, `brreg_snapshot()`, `brreg_snapshots()`

Examples

```
brreg_manifest()
```

`brreg_network` *Build an entity network graph*

Description

Construct a `tbl_graph` (tidygraph) representing the relationships around one or more seed entities. At depth 1, the graph includes sub-units, child entities, role holders, and legal role targets — all reachable via direct API calls. At depth 2, the graph expands through person nodes to discover board interlocks, requiring local bulk data (see Details).

Usage

```
brreg_network(
  org_nr,
  depth = 1L,
  include = c("underenheter", "children", "roles", "legal_roles"),
  download = FALSE
)
```

Arguments

org_nr	Character vector of seed organization numbers.
depth	Integer. 0 = seed only, 1 = ego network (default), 2 = expand through persons (requires bulk data).
include	Character vector of relationship types to include. Default includes all available types. Current types: "underenheter", "children", "roles", "legal_roles".
download	Logical. If TRUE and depth > 1, offer to download missing bulk data interactively. Default FALSE.

Value

A tbl_graph with node attributes node_id, node_type, name, org_nr, person_id, and edge attributes from, to, edge_type, role_code, role.

Depth and data requirements

- **Depth 0**: Seed entity only. 1 API call per seed.
- **Depth 1**: Full ego network. 5-7 API calls per seed.
- **Depth 2**: Board interlocks via person-to-entity reverse lookup. Requires local bulk data for enheter, underenheter, and roller. Run brreg_snapshot() for each type first, or call with download = TRUE to trigger downloads interactively.

Extensibility

The include parameter controls which relationship types are traversed. Each type maps to an internal collector function. Future versions may add types such as "addresses", "prior_owners", or "accounting".

See Also

brreg_entity() for single lookups, brreg_board_network() for the roles-only subgraph, brreg_status() to check bulk data availability.

Other tidybrreg governance functions: brreg_board_network(), brreg_survival_data()

Examples

```
net <- brreg_network("923609016")
net

tidygraph::as_tibble(net, "nodes")
tidygraph::as_tibble(net, "edges")
```

brreg_open *Open the snapshot store as a lazy Arrow Dataset*

Description

Returns an Arrow Dataset with Hive-style partitioning on `snapshot_date`. No data is loaded until `dplyr::collect()`. Requires the arrow package.

Usage

```
brreg_open(type = c("enheter", "underenheter", "roller"))
```

Arguments

`type`	One of `"enheter"` or `"underenheter"`.

Value

An `arrow::Dataset` object.

See Also

`brreg_panel()` for the higher-level panel constructor.

Other tidybrreg snapshot functions: `brreg_cleanup()`, `brreg_data_dir()`, `brreg_import()`, `brreg_manifest()`, `brreg_snapshot()`, `brreg_snapshots()`

Examples

```
ds <- brreg_open()
ds
```

brreg_panel *Construct a firm-period panel from accumulated snapshots*

Description

Build an unbalanced firm × period panel from the Hive-partitioned snapshot store. For each target period, selects the most recent prior snapshot (LOCF). Requires at least two snapshots saved via `brreg_snapshot()` or `brreg_import()`.

Usage

```
brreg_panel(
  frequency = c("year", "quarter", "month", "custom"),
  cols = NULL,
  from = NULL,
  to = NULL,
  dates = NULL,
  max_gap = NULL,
  type = c("enheter", "underenheter", "roller"),
  label = FALSE
)
```

Arguments

frequency	One of "year" (default), "quarter", "month", or "custom". For "custom", supply target dates via dates.
cols	Character vector of column names to include. NULL (default) returns all columns mapped by field_dict.
from, to	Start and end dates for the panel. NULL defaults to the range of available snapshots.
dates	Date vector for frequency = "custom".
max_gap	Integer. Maximum number of periods a snapshot may carry forward via LOCF. NULL (default) carries forward indefinitely. Set max_gap = 2 to prevent a quarterly snapshot from representing a firm as active 3+ quarters after its last observation.
type	One of "enheter" or "underenheter".
label	Logical. If TRUE, apply brreg_label() to the result.

Value

A tibble with columns: org_nr, period (character label for the period), snapshot_date (the actual snapshot used), plus requested cols. Attribute date_mapping records which snapshot was used for each period.

See Also

brreg_snapshot() to accumulate snapshots, brreg_events() for change detection between snapshots.

Other tidybrreg panel functions: as_brreg_tsibble(), brreg_change_summary(), brreg_changes(), brreg_events(), brreg_flows(), brreg_replay(), brreg_series()

Examples

```
# Annual panel of all entities
panel <- brreg_panel()
```

```
# Monthly panel for specific columns
panel <- brreg_panel("month", cols = c("employees", "nace_1", "legal_form"))
```

`brreg_replay` *Reconstruct register state by replaying CDC updates*

Description

Given a base snapshot and a stream of CDC updates from `brreg_updates()`, reconstruct the state of the register at any arbitrary date. Uses `dplyr::rows_upsert()` for Ny/Endring events and `dplyr::rows_delete()` for Sletting events, applied chronologically.

Usage

```
brreg_replay(base, updates, target_date = Sys.Date(), cols = NULL)
```

Arguments

`base`	A tibble from `brreg_download()`, `brreg_import()`, or a snapshot read via `brreg_open()`. Must contain `org_nr`.
`updates`	A tibble from `brreg_updates()` with `include_changes = TRUE`. Must contain `org_nr`, `change_type`, `timestamp`, and `changes` (list-column of patch tibbles).
`target_date`	Date. Reconstruct state as of this date. Only updates with `timestamp <= target_date` are applied.
`cols`	Character vector of columns to track. If `NULL`, all columns present in `base` are used.

Value

A tibble with the same columns as `base`, reflecting all applied changes up to `target_date`. Attribute `replay_info` records the number of inserts, updates, and deletes applied.

Limitations

The brreg CDC stream provides only new values (not old values) in RFC 6902 JSON Patch format, and field-level changes are only available from September 2025. Before that date, only the change type (Ny/Endring/Sletting) is recorded — field-level replay is not possible for those periods. Use `brreg_events()` (snapshot diff) for pre-September 2025 field-level changes.

See Also

`brreg_panel()` for multi-snapshot panels, `brreg_updates()` to fetch the CDC stream.

Other tidybrreg panel functions: `as_brreg_tsibble()`, `brreg_change_summary()`, `brreg_changes()`, `brreg_events()`, `brreg_flows()`, `brreg_panel()`, `brreg_series()`

Examples

```
base <- brreg_download(type_output = "tibble")
updates <- brreg_updates(since = Sys.Date() - 30,
                         size = 10000, include_changes = TRUE)
state <- brreg_replay(base, updates, target_date = Sys.Date())
```

`brreg_roles` *Retrieve board members, officers, and auditors*

Description

Fetch all registered roles for a Norwegian legal entity. Returns one row per role assignment. Person-held roles include name and birth date. Entity-held roles (auditor firms, accountants) include the entity's organization number.

Usage

```
brreg_roles(org_nr)
```

Arguments

`org_nr`	Character. 9-digit organization number.

Details

Role types and groups are returned as English labels looked up from the package's role_types and role_groups reference datasets. Original Norwegian codes are preserved in `role_code` and `role_group_code`.

Value

A tibble with one row per role assignment. Columns: `org_nr`, `role_group`, `role_group_code`, `role`, `role_code`, `first_name`, `middle_name`, `last_name`, `birth_date`, `deceased`, `entity_org_nr`, `entity_name`, `resigned`, `deregistered`, `ordering`, `elected_by`, `group_modified`, `person_id`. Returns an empty tibble if the entity has no registered roles.

Person identification

The `person_id` column is a synthetic key composed of birth date, last name, first name, and middle name. It enables network analysis across companies but has a non-trivial collision risk for common Norwegian names sharing a birth date. The brreg public API does not expose national identity numbers.

See Also

brreg_board_summary() for derived board covariates, role_types and role_groups for the English lookup tables.

Other tidybrreg entity functions: brreg_board_summary(), brreg_children(), brreg_download(), brreg_entity(), brreg_roles_legal(), brreg_search(), brreg_underenheter(), brreg_update_fields(), brreg_updates()

Examples

```
brreg_roles("923609016")  # Equinor ASA
```

---

brreg_roles_legal *Retrieve roles an entity holds in other entities*

Description

Reverse role lookup: find all entities where the given entity holds a role (e.g. parent company, shareholder, general partner). This is distinct from brreg_roles(), which returns who holds roles IN the given entity.

Usage

```
brreg_roles_legal(org_nr)
```

Arguments

org_nr	Character. 9-digit organization number.

Value

A tibble with one row per role held. Columns: org_nr (queried entity), target_org_nr (entity where role is held), target_name, role_code, role, share (ownership share if applicable), resigned, deregistered.

See Also

brreg_roles() for who holds roles in an entity.

Other tidybrreg entity functions: brreg_board_summary(), brreg_children(), brreg_download(), brreg_entity(), brreg_roles(), brreg_search(), brreg_underenheter(), brreg_update_fields(), brreg_updates()

Examples

```
brreg_roles_legal("923609016")  # Equinor's roles in other entities
```

brreg_search *Search Norwegian legal entities*

Description

Query the Central Coordinating Register by name, legal form, industry, geography, and other criteria. Results are paginated automatically up to `max_results` or the API's 10,000-result ceiling, whichever is lower.

Usage

```
brreg_search(
  name = NULL,
  legal_form = NULL,
  municipality_code = NULL,
  nace_code = NULL,
  min_employees = NULL,
  max_employees = NULL,
  bankrupt = NULL,
  parent_org_nr = NULL,
  max_results = 200,
  registry = c("enheter", "underenheter"),
  type = c("code", "label")
)
```

Arguments

`name`	Character. Entity name (partial match, case-insensitive).
`legal_form`	Character. Legal form code: `"AS"`, `"ASA"`, `"ENK"`, etc. See legal_forms for valid codes.
`municipality_code`	Character. 4-digit Norwegian municipality code.
`nace_code`	Character. NACE industry code (e.g. `"64.190"`).
`min_employees, max_employees`	Integer. Employee count range.
`bankrupt`	Logical. If `TRUE`, return only bankrupt entities.
`parent_org_nr`	Character. Filter to subsidiaries of this org.
`max_results`	Integer. Maximum entities to return (default 200). The API caps search results at 10,000; use `brreg_download()` for larger extractions.
`registry`	One of `"enheter"` (main entities, default) or `"underenheter"` (sub-entities / establishments). Sub-entities use `beliggenhetsadresse.kommunenummer` instead of `kommunenummer` for geographic filtering, and the `bankrupt` parameter is not available.
`type`	A type of variables: `"code"` (default) returns raw codes, `"label"` returns English labels for coded columns.

Value

A tibble with one row per entity. Column names follow the package field dictionary (field_dict). An attribute `total_matches` records the total number of matches in the registry.

Norwegian legal forms

Common codes for the `legal_form` parameter:

- **AS**: Private limited company (like UK Ltd, German GmbH)
- **ASA**: Public limited company (like UK PLC, German AG)
- **ENK**: Sole proprietorship
- **NUF**: Norwegian-registered foreign entity (branch office)

See legal_forms for the complete list with English translations.

See Also

`brreg_entity()` for single lookups, `brreg_label()` for translating codes to English.

Other tidybrreg entity functions: `brreg_board_summary()`, `brreg_children()`, `brreg_download()`, `brreg_entity()`, `brreg_roles()`, `brreg_roles_legal()`, `brreg_underenheter()`, `brreg_update_fields()`, `brreg_updates()`

Examples

```
# Search by name
brreg_search(name = "Equinor")

# Large private companies in Oslo
brreg_search(legal_form = "AS", municipality_code = "0301",
             min_employees = 500, max_results = 10)
```

`brreg_series` *Compute aggregate time series from snapshots*

Description

Produce period-level summary statistics from the snapshot store for any combination of variables and summary functions. Returns a tibble suitable for ggplot2 or `as_brreg_tsibble()` conversion.

Usage

```
brreg_series(
  .vars = NULL,
  .fns = list(total = function(x) sum(x, na.rm = TRUE)),
  by = NULL,
  frequency = c("year", "quarter", "month"),
  from = NULL,
  to = NULL,
  type = c("enheter", "underenheter", "roller"),
  label = FALSE
)
```

Arguments

.vars	Character vector of column names to aggregate. NULL (default) counts entities per period.
.fns	Named list of summary functions applied to each column in .vars. Default: list(total = \(x) sum(x, na.rm = TRUE)). Use list(avg = mean, sd = sd) for multiple summaries. Output columns are named {variable}_{function}.
by	Character vector of grouping column names (e.g. "nace_1", c("legal_form", "municipality_code")). NULL for national totals.
frequency	One of "year", "quarter", "month".
from, to	Start and end dates for the panel. NULL defaults to the range of available snapshots.
type	One of "enheter", "underenheter", "roller".
label	Logical. Translate group codes to English labels.

Value

A tibble with period (character), optional grouping columns, and one column per variable-function combination. Attribute brreg_panel_meta records metadata for as_brreg_tsibble() conversion.

See Also

brreg_panel() for entity-level panels, as_brreg_tsibble() for tsibble conversion.

Other tidybrreg panel functions: as_brreg_tsibble(), brreg_change_summary(), brreg_changes(), brreg_events(), brreg_flows(), brreg_panel(), brreg_replay()

Examples

```
brreg_series(.vars = "employees", by = "legal_form")

brreg_series(.vars = "employees",
             .fns = list(avg = mean, total = sum),
             by = "nace_1")
```

brreg_snapshot *Save a dated snapshot of the full register*

Description

Download today's complete register and save as a Parquet partition in the local snapshot store. Each call adds one partition to a Hive-partitioned dataset at `tools::R_user_dir("tidybrreg", "data")/{type}/snapshot_da`. Subsequent calls to `brreg_panel()` and `brreg_events()` query this partitioned dataset lazily via `arrow::open_dataset()`.

Usage

```
brreg_snapshot(
  type = c("enheter", "underenheter", "roller"),
  format = c("csv", "json"),
  date = Sys.Date(),
  force = FALSE,
  ask = interactive()
)
```

Arguments

`type`	One of `"enheter"` (main entities, default), `"underenheter"` (sub-entities / establishments), or `"roller"` (all roles for all entities, via `/roller/totalbestand`). Roller snapshots parse the nested JSON into a flat tibble matching `brreg_roles()` output.
`format`	Download format: `"csv"` (default for enheter/underenheter) or `"json"`. JSON captures additional fields not present in CSV (e.g. `kapital`, `vedtektsfestetFormaal`, `paategninger`). Roller is always JSON regardless of this parameter.
`date`	Date for this snapshot (default: today). Used as the partition key, not as an API parameter — the brreg bulk endpoint always returns the current-day state.
`force`	Logical. If `TRUE`, overwrite an existing partition for this date. Default `FALSE` skips if partition exists.
`ask`	Logical. If `TRUE` (the default in interactive sessions), prompt before downloading ~145 MB.

Value

The file path to the written Parquet partition (invisibly).

See Also

brreg_import() to add historical snapshots from CSV files, brreg_snapshots() to list available snapshots, brreg_panel() to construct panels from accumulated snapshots.

Other tidybrreg snapshot functions: brreg_cleanup(), brreg_data_dir(), brreg_import(), brreg_manifest(), brreg_open(), brreg_snapshots()

Examples

```
brreg_snapshot()
brreg_snapshots()
```

---

brreg_snapshots *List available snapshots*

Description

Scan the local snapshot store and return metadata for each partition.

Usage

```
brreg_snapshots(type = c("enheter", "underenheter", "roller"))
```

Arguments

type	One of "enheter" or "underenheter".

Value

A tibble with columns: snapshot_date (Date), file_size (numeric, bytes), path (character).

See Also

Other tidybrreg snapshot functions: brreg_cleanup(), brreg_data_dir(), brreg_import(), brreg_manifest(), brreg_open(), brreg_snapshot()

Examples

```
brreg_snapshots()
```

brreg_status *Check availability of local bulk datasets*

Description

Inspects the snapshot store and download cache for each requested dataset type. Used internally by `brreg_network()` to gate depth > 1 operations that require local data for person-to-entity reverse lookups.

Usage

```
brreg_status(datasets = c("enheter", "underenheter", "roller"), quiet = FALSE)
```

Arguments

`datasets`	Character vector of dataset types to check.
`quiet`	Logical. If `TRUE`, suppress informational messages.

Value

A list with components: `available` (character vector of datasets found locally), `missing` (character vector of datasets not found), `all_ready` (logical).

See Also

`brreg_snapshot()` to download and cache bulk data, `brreg_download()` for one-off downloads.

Other tidybrreg data management functions: `brreg_annotation_summary()`, `brreg_annotations()`, `brreg_sync()`, `brreg_sync_status()`, `diff_roller_state()`, `read_changelog()`

Examples

```
brreg_status()
```

brreg_survival_data *Prepare firm survival data*

Description

Compute time-to-event and censoring indicators from entity registration data, ready for use with `survival::Surv()` and `flexsurv`.

Usage

```
brreg_survival_data(
  data,
  entry_var = "founding_date",
  censoring_date = Sys.Date()
)
```

Arguments

data	A tibble from brreg_download(), brreg_search(), or brreg_panel(). Must contain at least org_nr and a date column for entry.
entry_var	Column name for the entry date. Default "founding_date" (stiftelsesdato).
censoring_date	Date at which surviving firms are right-censored. Default: today.

Value

A tibble with added columns: entry_date, exit_date (Date or NA), duration_years (numeric), event (integer: 1 = exit observed, 0 = right-censored). Compatible with survival::Surv(duration_years, event).

See Also

brreg_download() for full register data.

Other tidybrreg governance functions: brreg_board_network(), brreg_network()

Examples

```
firms <- brreg_search(legal_form = "AS", municipality_code = "0301",
                      max_results = 100)
surv <- brreg_survival_data(firms)
surv[, c("org_nr", "entry_date", "exit_date", "duration_years", "event")]
```

---

brreg_sync *Synchronize local state with the brreg CDC stream*

Description

Maintains a local mirror of the Enhetsregisteret by applying incremental CDC (change data capture) events to a persistent state table. On first run, bootstraps from a bulk download. Subsequent runs poll the CDC endpoints from the last cursor position and apply mutations.

Usage

```
brreg_sync(
  types = c("enheter", "underenheter", "roller"),
  size = 10000L,
  roller_method = c("bulk", "cdc"),
  verbose = TRUE
)
```

Arguments

types	Character vector of streams to sync. Default syncs all four.
size	Integer. CDC page size per API call (max 10000).
roller_method	One of "bulk" (default) or "cdc". "bulk" downloads the full totalbestand (~131 MB) and computes a field-level diff against previous state — fast and produces granular changelogs. "cdc" fetches current roles per-org via the API for each CDC event — slower but provides sub-daily attribution when syncing multiple times per day.
verbose	Logical. Print progress messages.

Details

Four state files are maintained:

- enheter.parquet — main entities (~1M rows)
- underenheter.parquet — sub-entities (~500K rows)
- roller.parquet — all roles (~3.4M rows)
- paategninger.parquet — registry annotations

Every mutation is logged to a Hive-partitioned changelog under state/changelog/sync_date={date}/. The changelog drives brreg_changes() and brreg_flows().

Value

A list with sync summary: events processed per type, changelog rows written, elapsed time.

Write ordering

Changelog is written first (WAL), then state files, then cursor. If a crash occurs between state and cursor, the next sync replays from the old cursor. Mutations are idempotent (upsert by org_nr), so replay is safe.

See Also

brreg_sync_status() to check current state, brreg_changes() to query the changelog, brreg_flows() for entry/exit counts.

Other tidybrreg data management functions: brreg_annotation_summary(), brreg_annotations(), brreg_status(), brreg_sync_status(), diff_roller_state(), read_changelog()

Examples

```
brreg_sync()
brreg_sync_status()
```

---

brreg_sync_status *Display sync status*

Description

Shows the current state of the sync engine: which state files exist, when the last sync occurred, cursor positions, and changelog size.

Usage

```
brreg_sync_status()
```

Value

A list with status components (invisibly).

See Also

`brreg_sync()` to run a sync, `brreg_changes()` to query the changelog.

Other tidybrreg data management functions: `brreg_annotation_summary()`, `brreg_annotations()`, `brreg_status()`, `brreg_sync()`, `diff_roller_state()`, `read_changelog()`

Examples

```
brreg_sync_status()
```

---

brreg_underenheter *Get all sub-units (underenheter) belonging to an entity*

Description

Convenience wrapper around `brreg_search()` that queries the underenheter registry filtered by `overordnetEnhet`. Each Norwegian legal entity that operates a business has one or more sub-units (BEDR/AAFY) representing physical locations or activities.

Usage

```
brreg_underenheter(org_nr, max_results = 200, type = c("code", "label"))
```

Arguments

org_nr	Character. 9-digit organization number of the parent entity (hovedenhet).
max_results	Integer. Maximum sub-units to return (default 200).
type	One of "code" (default) or "label".

Value

A tibble with one row per sub-unit. Same column schema as brreg_search() with registry = "underenheter".

See Also

brreg_entity() for the parent entity, brreg_children() for child enheter in the ORGL hierarchy.

Other tidybrreg entity functions: brreg_board_summary(), brreg_children(), brreg_download(), brreg_entity(), brreg_roles(), brreg_roles_legal(), brreg_search(), brreg_update_fields(), brreg_updates()

Examples

```
brreg_underenheter("923609016")
```

---

brreg_update_fields *Retrieve field-level CDC changes as a flat tibble*

Description

Returns one row per field-level change, plus one synthetic row for each event that carries no field patches (Ny, Sletting, Fjernet). Synthetic rows have operation = NA, field = NA, new_value = NA, preserving event-level metadata in the output.

Usage

```
brreg_update_fields(
  since = Sys.Date() - 1,
  size = 100,
  max_pages = 1L,
  type = c("enheter", "underenheter"),
  verbose = FALSE
)
```

Arguments

since	Date or POSIXct. Return updates after this timestamp. Defaults to yesterday.
size	Integer. Number of updates per page (max 10000).
max_pages	Integer. Maximum pages to fetch. Default 1. Set higher to paginate through large result sets automatically.
type	One of "enheter", "underenheter", or "roller". Roller uses CloudEvents format; include_changes is ignored.
verbose	Logical. Print page-level progress.

Details

To retrieve initial field values for newly registered entities (change_type "Ny"), call brreg_entity() per org_nr — the CDC payload for Ny events contains no field-level patches.

Value

A tibble with columns: update_id, org_nr, change_type, timestamp, operation, field, new_value. No list-columns. Events without field patches (Ny, Sletting, Fjernet) appear as rows with operation, field, and new_value all NA.

See Also

brreg_updates() for the event-level view.

Other tidybrreg entity functions: brreg_board_summary(), brreg_children(), brreg_download(), brreg_entity(), brreg_roles(), brreg_roles_legal(), brreg_search(), brreg_underenheter(), brreg_updates()

Examples

```
brreg_update_fields(since = Sys.Date() - 1, size = 50)
```

---

brreg_updates *Retrieve incremental entity updates*

Description

Query the brreg change data capture (CDC) endpoint for entities modified since a given date. Each update carries a monotonically increasing update_id suitable for cursor-based pagination and deduplication.

Usage

```
brreg_updates(
  since = Sys.Date() - 1,
  size = 100,
  max_pages = 1L,
  include_changes = FALSE,
  type = c("enheter", "underenheter", "roller"),
  verbose = FALSE
)
```

Arguments

since	Date or POSIXct. Return updates after this timestamp. Defaults to yesterday.
size	Integer. Number of updates per page (max 10000).
max_pages	Integer. Maximum pages to fetch. Default 1. Set higher to paginate through large result sets automatically.
include_changes	Logical. If TRUE, include field-level change details per update as a list-column of tibbles.
type	One of "enheter", "underenheter", or "roller". Roller uses CloudEvents format; include_changes is ignored.
verbose	Logical. Print page-level progress.

Value

A tibble with columns: update_id, org_nr, change_type, timestamp. If include_changes = TRUE, a list-column changes with tibbles of operation, field, new_value.

See Also

brreg_update_fields() for a flat alternative.

Other tidybrreg entity functions: brreg_board_summary(), brreg_children(), brreg_download(), brreg_entity(), brreg_roles(), brreg_roles_legal(), brreg_search(), brreg_underenheter(), brreg_update_fields()

Examples

```
brreg_updates(since = Sys.Date() - 1, size = 10)

# Auto-paginate
brreg_updates(since = "2026-03-01", size = 10000, max_pages = 50,
              verbose = TRUE)
```

brreg_validate *Validate Norwegian organization numbers*

Description

Check whether organization numbers (organisasjonsnummer) pass the modulus-11 check digit algorithm used by Norway's Central Coordinating Register for Legal Entities. Valid numbers are exactly 9 digits, start with 8 or 9, and have a correct check digit computed with weights `3, 2, 7, 6, 5, 4, 3, 2`.

Usage

```
brreg_validate(org_nr)
```

Arguments

`org_nr`	Character vector of organization numbers to validate.

Value

Logical vector the same length as `org_nr`. `TRUE` for valid numbers, `FALSE` otherwise.

See Also

`brreg_entity()` which validates before querying the API.

Other tidybrreg utilities: `brreg_label()`, `get_brreg_dic()`

Examples

```
brreg_validate(c("923609016", "984851006", "123456789", "999999999"))
```

diff_roller_state *Compute field-level diffs between two roller state tibbles*

Description

Pure function: takes two flattened roller tibbles (as produced by `flatten_roles()` or `parse_roles_bulk()`) and returns a long-format changelog recording every field-level mutation. Detects three categories of change: role additions, role removals, and field-level modifications on continuing roles.

Usage

```
diff_roller_state(
  old_state,
  new_state,
  timestamp = format(Sys.time(), "%Y-%m-%dT%H:%M:%S"),
  update_id = NA_integer_
)
```

Arguments

old_state	Tibble. Previous roller state from read_state() or an earlier flatten_roles() call. NULL or zero-row tibble treats all current roles as additions.
new_state	Tibble. Current roller state from brreg_download() or flatten_roles().
timestamp	Character or POSIXct. Timestamp for changelog entries (typically the CDC event time or sync time).
update_id	Integer or character. Identifier for the sync batch, used as update_id in the changelog.

Value

A tibble matching the changelog schema: timestamp, org_nr, registry (always "roller"), change_type ("entry", "exit", "change"), field, value_from, value_to, update_id.

Composite key

Each role assignment is identified by (org_nr, role_group_code, role_code, holder_id) where holder_id is derived from person_id for person-held roles and entity:{entity_org_nr} for entity-held roles (auditors, accountants). Roles with neither are keyed as unknown:{row_index} within their respective state — these are rare and produce conservative add/remove pairs rather than false modifications.

NA handling

For additions, fields that are NA in the new state are excluded from the changelog (no value to report). For removals, fields that are NA in the old state are excluded. For modifications, a change from NA to a non-NA value (or vice versa) is recorded.

See Also

brreg_sync() for automated sync with changelog persistence, brreg_roles() for fetching current roles, brreg_changes() for querying stored changelogs.

Other tidybrreg data management functions: brreg_annotation_summary(), brreg_annotations(), brreg_status(), brreg_sync(), brreg_sync_status(), read_changelog()

Examples

```
# Detect board changes for a single company
old <- brreg_roles("923609016")
Sys.sleep(1)
new <- brreg_roles("923609016")
diff_roller_state(old, new)

# Bootstrap: NULL old state treats all roles as entries
roles <- brreg_roles("923609016")
diff_roller_state(NULL, roles)
```

field_dict *Field dictionary: Norwegian API paths to English column names*

Description

A tibble mapping brreg JSON field paths to English column names and R types. Used internally by the parse engine. API fields absent from this dictionary pass through with auto-generated snake_case names rather than being silently dropped.

Usage

```
field_dict
```

Format

A tibble with 103 rows and 3 columns:

api_path Dot-notation path in the brreg JSON response (e.g. `"organisasjonsnummer"`, `"forretningsadresse.kommune"`)

col_name English column name used in package output (e.g. `"org_nr"`, `"municipality"`).

type R type for coercion: `"character"`, `"Date"`, `"integer"`, `"numeric"`, or `"logical"`.

See Also

`brreg_entity()` for the function that uses this dictionary.

Other tidybrreg reference data: `annotation_infotypes`, `legal_forms`, `role_groups`, `role_types`

Examples

```
field_dict
field_dict[field_dict$type == "Date", ]
```

get_brreg_dic *Fetch a brreg dictionary*

Description

Retrieve English or Norwegian label dictionaries for NACE industry codes or institutional sector codes. Dictionaries are cached in a session-level environment (following the eurostat package pattern). Bundled data is used as fallback when the SSB Klass API is unreachable.

Usage

```
get_brreg_dic(dictname = c("nace", "sector"), lang = "en")
```

Arguments

dictname	One of "nace" or "sector".
lang	"en" (default) for English labels or "no" for Norwegian.

Value

A tibble with columns code, name_en, level.

See Also

brreg_label() for translating columns in place.

Other tidybrreg utilities: brreg_label(), brreg_validate()

Examples

```
get_brreg_dic("nace")
get_brreg_dic("sector")
```

---

legal_forms *Norwegian legal form codes with English translations*

Description

All organisasjonsformer registered with the Brønnøysund Register Centre, with English translations. Fetched from the brreg API during package build and supplemented with manual English translations.

Usage

```
legal_forms
```

Format

A tibble with 44 rows and 4 columns:

code Legal form code (e.g. "AS", "ASA", "ENK").

name_no Norwegian description.

expired Date string if the form is expired, NA otherwise.

name_en English translation.

See Also

brreg_label() to translate legal form codes in entity data.

Other tidybrreg reference data: annotation_infotypes, field_dict, role_groups, role_types

Examples

```
legal_forms
legal_forms[legal_forms$code == "AS", ]
```

`role_groups` *Role group codes with English translations*

Description

Maps brreg rollegruppe codes to English names.

Usage

```
role_groups
```

Format

A tibble with 15 rows and 3 columns:

code Role group code (e.g. `"STYR"`, `"DAGL"`, `"REVI"`).

name_en English translation (e.g. `"Board of Directors"`).

name_no Norwegian description.

See Also

`brreg_roles()`, role_types.

Other tidybrreg reference data: `annotation_infotypes`, `field_dict`, `legal_forms`, `role_types`

Examples

```
role_groups
```

`role_types` *Role type codes with English translations*

Description

Maps brreg rolle codes to English names. Used by `brreg_roles()` for automatic role labelling and by `brreg_label()` for post-hoc translation.

Usage

```
role_types
```

Format

A tibble with 18 rows and 3 columns:

code Role code (e.g. `"LEDE"`, `"MEDL"`, `"DAGL"`).

name_en English translation (e.g. `"Chair of the Board"`).

name_no Norwegian description.

See Also

`brreg_roles()`, role_groups.

Other tidybrreg reference data: `annotation_infotypes`, `field_dict`, `legal_forms`, `role_groups`

Examples

```
role_types
```

Index