Package: tidybrreg (via r-universe)

November 2, 2026

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

Version 0.6.0

Description A tidy interface to query and download data from the Norwegian Central Coordinating Register for Legal Entities (Enhetsregisteret), maintained by the Brønnøysund Register Centre. Looks up entities, sub-entities, roles, corporate groups and signing authority, searches the register, reads the change data capture feeds, and downloads the bulk register. Returns tibbles with English column names mapped via a data-driven dictionary; unknown API fields pass through with auto-generated names, and only observed fields are emitted. Coded values (legal forms, industry codes, roles) can be translated to English or Norwegian using bundled reference data or live lookups from the Statistics Norway 'KLASS' API. Register data is published 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, rlang, tibble, vctrs

Suggests callr, curl, knitr, pkgload, rmarkdown, testthat (>= 3.0.0), tidyr, withr, yyjsonr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.1

Config/pak/sysreqs libssl-dev

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

Date/Publication 2026-11-02 18:57:27 UTC

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

RemoteRef HEAD

RemoteSha 55e87e08a95137ad97098a794e327be25ac6aa01

Contents

brreg_children . . . 3
brreg_download . . . 4
brreg_entity . . . 6
brreg_konsern . . . 8
brreg_label . . . 9
brreg_prokura . . . 11
brreg_regnskap . . . 12
brreg_roles . . . 15
brreg_roles_legal . . . 16
brreg_search . . . 17
brreg_signatur . . . 20
brreg_underenheter . . . 21
brreg_update_fields . . . 22
brreg_updates . . . 24
brreg_validate . . . 26
field_dict . . . 26
get_brreg_dic . . . 27
legal_forms . . . 28
regnskap_dict . . . 29
regnskap_groups . . . 30
regnskap_items . . . 31
role_groups . . . 32
role_types . . . 32
tidybrreg_conditions . . . 33
tidybrreg_endpoints . . . 35
tidybrreg_provenance . . . 36

Index **38**

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 and from the konsern ownership tree returned by `brreg_konsern()`.

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). When the parent has more, the result carries `complete = FALSE` and a `tidybrreg_warning_truncated` is signalled.
`type`	One of `"code"` (default) or `"label"`.

Value

A tibble with one row per child entity. Same column schema and attributes 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_download()`, `brreg_entity()`, `brreg_konsern()`, `brreg_prokura()`, `brreg_regnskap()`, `brreg_roles()`, `brreg_roles_legal()`, `brreg_search()`, `brreg_signatur()`, `brreg_underenheter()`, `brreg_update_fields()`, `brreg_updates()`

Examples

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

brreg_download *Download the full Norwegian business register*

Description

Download a complete extract of the Central Coordinating Register for Legal Entities (~1 million entities, ~150 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"),
  refresh = FALSE,
  cache = TRUE,
  type_output = c("tibble", "path")
)
```

Arguments

`type`	One of `"enheter"` (main entities, default, ~196 MB), `"underenheter"` (sub-entities / establishments, ~75 MB), or `"roller"` (all roles for all entities via `/roller/totalbestand`, ~131 MB). All three are gzipped JSON.
`refresh`	`FALSE` (default): use cached file if available. `TRUE`: force re-download. `"auto"`: revalidate against the server and re-download only when the cached copy cannot be shown current. Any other value is an error.
`cache`	Logical. If `TRUE` (default), keep the downloaded file in the persistent cache directory; if `FALSE`, use a temporary file.
`type_output`	One of `"tibble"` (default) 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; arrays of objects are list-columns of tibbles. For `type = "roller"`, one row per role assignment with the columns of `brreg_roles()`.
- `"path"`: Character file path to the gzipped download.

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 JSON. Always returns the full register. Appropriate for any analysis requiring more than 10,000 entities.

Results from both paths use the same column names and types via field_dict; only observed columns are emitted (see brreg_entity()). The bulk is the JSON variant of each endpoint, the same representation the single-entity lookups and the update feeds use, so historiskeNavn and paategninger arrive as the nested list-columns historiske_navn and annotations exactly as from brreg_entity(). BRREG's CSV variant is not read: it is the register's own flattening, with different separators and a presence flag in place of the annotation array.

Memory and time

The tibble of the full register is built in memory. The gzipped array is decoded in batches of complete objects and the batches are bound one column at a time, so the peak is the result plus the decoder's transient rather than a multiple of the result. Measured on the live register on 2026-11-02 (2 cores): enheter, 1,179,853 rows and 91 columns, 14.6 minutes, a 1.6 GB tibble, 4.0 GB peak resident memory; underenheter, 870,571 rows and 43 columns, 7.1 minutes, 0.9 GB, 2.0 GB peak; roller, 3,426,438 roles of 1,154,321 entities, 11.1 minutes, 0.7 GB, 2.3 GB peak with jsonlite (yyjsonr, when installed, decodes each batch faster). A machine with 8 GB holds one full enheter parse and not two at once. type_output = "path" costs nothing beyond the download, and parse_bulk_json(path, n_max =) (internal) reads the head of the array without inflating the rest.

Caching

With cache = TRUE (default) the gzipped payload is stored in tools::R_user_dir("tidybrreg", "cache") together with a sidecar recording the server's ETag and Last-Modified validators, and a later call reuses it. refresh = TRUE forces a new download. refresh = "auto" sends a HEAD request and re-downloads when either validator differs from the sidecar; when the sidecar is missing or the server sends neither validator, nothing can prove the cache current, so it is re-downloaded. With cache = FALSE the payload goes to a temporary file that is deleted after parsing; with type_output = "path" that temporary path is returned and the caller owns it. The cache directory is created only by this function, never at load time.

Metadata of the most recent download (url, etag, last_modified as a UTC POSIXct, downloaded_at, path) is available as attr(result, "brreg_download") on tibble output.

Tidy shape

Every tibble the package returns is tidy in Wickham's sense: one observation per row, one variable per column, one observational unit per table. The register nests two kinds of array inside an entity. Arrays of strings (business_address, postal_address, location_address, purpose, activity) are the line fragments of one value and are joined with "\n", which no fragment contains, so strsplit(x, "\n") recovers the lines. Arrays of objects (historiske_navn, annotations, any array of objects the register adds) are observations of another unit and are nested as a list-column of tibbles, one row per element with snake_case columns, NULL for an entity without them; tidyr::unnest() expands them to their own table. An empty array is not an observation and emits no column. Nothing is serialized to JSON text.

See Also

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

Other tidybrreg entity functions: brreg_children(), brreg_entity(), brreg_konsern(), brreg_prokura(), brreg_regnskap(), brreg_roles(), brreg_roles_legal(), brreg_search(), brreg_signatur(), brreg_underenheter(), brreg_update_fields(), brreg_updates()

Examples

```
# Download full register as tibble (~196 MB 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 single 9-digit Norwegian organization number. Validated using brreg_validate() before the API call; a vector of several numbers is an error (iterate with lapply()).
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. Labelling NACE and sector codes consults SSB KLASS on first use in a session; see get_brreg_dic().

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 field the API returned, named via field_dict, plus registry (which registry answered) and deleted (logical). Frequently present columns include org_nr, name, legal_form, employees, founding_date, nace_1, municipality_code, bankrupt, and parent_org_nr; historiske_navn and annotations are list-columns of tibbles (see the section on tidy shape).

Observed fields only

The register emits only the fields that apply to an entity: a sole proprietorship (ENK) typically carries no founding_date or employees, a foreign branch (NUF) no municipality_code. The returned columns are exactly the fields the response carried, so every column other than org_nr, registry and deleted is conditionally present. Test for a column with "x" %in% names(result) rather than assuming it.

Tidy shape

Every tibble the package returns is tidy in Wickham's sense: one observation per row, one variable per column, one observational unit per table. The register nests two kinds of array inside an entity. Arrays of strings (business_address, postal_address, location_address, purpose, activity) are the line fragments of one value and are joined with "\n", which no fragment contains, so strsplit(x, "\n") recovers the lines. Arrays of objects (historiske_navn, annotations, any array of objects the register adds) are observations of another unit and are nested as a list-column of tibbles, one row per element with snake_case columns, NULL for an entity without them; tidyr::unnest() expands them to their own table. An empty array is not an observation and emits no column. Nothing is serialized to JSON text.

Deleted entities

BRREG reports a deleted entity either as HTTP 410 or, for most deletions, as HTTP 200 with respons_klasse = "SlettetEnhet" and a slettedato. Both are modelled the same way: deleted is TRUE, deletion_date is the parsed date, every other field the response carried is kept, and a tidybrreg_warning_deleted is signalled (silence it with suppressWarnings(classes = "tidybrreg_warning_deleted"
Live entities carry deleted = FALSE.

See Also

brreg_search() for querying multiple entities, brreg_label() for translating codes to English, field_dict for the column name mapping, tidybrreg_conditions for the errors and warnings signalled.

Other tidybrreg entity functions: brreg_children(), brreg_download(), brreg_konsern(), brreg_prokura(), brreg_regnskap(), brreg_roles(), brreg_roles_legal(), brreg_search(), brreg_signatur(), 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_konsern *Retrieve the corporate group structure (konsernstruktur)*

Description

Fetch the registered group hierarchy for a Norwegian legal entity from the Enhetsregisteret konsernstruktur endpoint (added by BRREG 2026-06-24). The tree is returned flat: one row per node, root included at `level = 0`, each child carrying its link form (`knytningsform`, e.g. `KDAT` "Konsern datter"), ownership basis (`grunnlag`, e.g. `"100%"`), and the registration date of the relation.

Usage

```
brreg_konsern(org_nr)
```

Arguments

org_nr	Character. A single 9-digit organization number.

Details

The JSON response omits `knytningsform` on the root node while the API's CSV variant emits it as `KKKK` "Øverste mor"; the root row is synthesized here to match the CSV semantics.

This is the ownership hierarchy and is distinct from brreg_children(), which searches `overordnetEnhet` — the administrative parent relation used in public-sector (ORGL) structures, not konsern ownership.

Value

A tibble with one row per node in the group tree. Columns: `org_nr` (the queried entity), `node_org_nr`, `node_name`, `level` (0 for the root), `link_form`, `link_form_desc`, `basis`, `relation_date` (Date), `parent_org_nr`, `parent_name`, `legal_form`, `legal_form_desc`. A zero-row tibble with the same typed columns if no group structure is registered.

404 semantics

The endpoint returns 404 both for organization numbers that do not exist and for live entities that are not part of any registered group — the common case for ordinary companies. The `in_corporate_group` column from `brreg_entity()` disambiguates the two without a second lookup here.

Status handling

The per-entity endpoints behind `brreg_roles()`, `brreg_roles_legal()`, `brreg_konsern()`, `brreg_signatur()`, `brreg_prokura()` and `brreg_regnskap()` share one contract. HTTP 404 means the organization number is unknown to that endpoint, which for konsern, signatur, prokura and regnskap is the ordinary case of an entity with nothing registered there: the typed zero-row tibble is returned with attribute `http_status = 404L` and no warning. HTTP 410 means the entity has been deleted: the typed zero-row tibble is returned with `http_status = 410L`, `deletion_date`, and a `tidybrreg_warning_deleted`. Any other status of 400 or above aborts with a `tidybrreg_error_http`. A successful response with nothing registered returns the typed zero-row tibble with no attribute. Use `brreg_entity()` to distinguish a non-existent organization number from a live entity with nothing registered.

See Also

`brreg_children()` for the ORGL administrative hierarchy, `brreg_entity()` for the `in_corporate_group` flag, `brreg_roles_legal()` for roles an entity holds in others.

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

Examples

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

---

brreg_label *Translate codes to human-readable labels*

Description

Replace coded values in a brreg tibble with English (or Norwegian) 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 = c("en", "no"))
```

Arguments

x	A tibble from brreg_entity(), brreg_search(), brreg_roles() or brreg_regnskap(), 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", "regnskap_item", "regnskap_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 before the label column. For example, brreg_label(x, code = "legal_form") adds legal_form_code.
lang	Language for labels: "en" (default) or "no" (Norwegian, as published by BRREG and SSB).

Value

When x is a data frame: the same tibble with code columns replaced by labels. When x is a character vector: a character vector of labels, without names.

Which columns are labelled

legal_form (from legal_forms), role_code (from role_types), role_group_code (from role_groups), nace_1, nace_2, nace_3 and sector_code (from the NACE and institutional-sector dictionaries of get_brreg_dic()), and the accounts columns item (from regnskap_items), group and statement (from regnskap_groups), inside a nested items list-column as well as on an unnested table. Other columns are never touched. Codes with no dictionary entry pass through unchanged, and labels are not codes, so applying brreg_label() twice is the same as applying it once. Attributes of the input (such as total_matches from brreg_search()) are preserved.

Network access

Legal-form and role dictionaries are bundled and never need the network. NACE and sector dictionaries come from the SSB KLASS API on first use in a session and are cached for the session; when KLASS cannot be reached, or options(tidybrreg.offline = TRUE) is set, the bundled English dictionaries are used instead and the result carries their vintage. See get_brreg_dic() for the full policy.

See Also

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

Other tidybrreg utilities: brreg_validate(), get_brreg_dic(), tidybrreg_conditions, tidybrreg_endpoints

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_label(c("AS", "ASA", "ENK"), dic = "legal_form", lang = "no")
brreg_label(c("total_assets", "net_income"), dic = "regnskap_item", lang = "no")
```

`brreg_prokura` *Retrieve procuration authority (prokura)*

Description

Fetch the registered procuration combinations for a Norwegian legal entity from the Brønnøysund Fullmakt service. Procuration is a commercial power of attorney under the Norwegian Powers of Attorney Act (prokuraloven): a holder may bind the entity in ordinary business, but not, absent separate authority, sell or encumber its real property. It is a narrower and separate mandate from the general signing authority returned by `brreg_signatur()`.

Usage

```
brreg_prokura(org_nr)
```

Arguments

`org_nr`	Character. A single 9-digit organization number.

Details

One row is returned per person within each registered combination.

Value

A tibble with the same columns as `brreg_signatur()`, with `signature_type` set to `"procuration"`. A zero-row tibble with the same typed columns if no procuration is registered.

Person identification

The Fullmakt service returns each person's name as a single string in `name`; it does not split it into given and family names, so no synthetic `person_id` is constructed. Join to `brreg_roles()` on `birth_date` together with `name` at query time if cross-referencing to the role network is required. Role designations are returned as English labels looked up from role_types, with the original Norwegian code preserved in `role_code`; the signing-mode designations (`SIGN`, `SIFE`, `SIHV`, `PROK`, `POFE`, `POHV`) are included in role_types alongside the board roles. Birth dates arrive as `DD.MM.YYYY` from this service and as ISO dates from the roles endpoint; both are parsed to `Date`, so the two join directly.

Rules the service declines to resolve

For some entities (Equinor ASA among them) the service answers HTTP 200 with no combinations and a routine status instead: code NA, "Generelle unntak, finnes roller som har status det ikke kan foretas maskinelt oppslag mot", meaning a rule is registered but cannot be resolved mechanically. The typed zero-row tibble is then returned with attributes `http_status = 200L`, `routine_status` and `routine_text`, and a `tidybrreg_warning_unresolved` is signalled, so the case is distinguishable from an entity with nothing registered (no attribute, no warning).

Status handling

The per-entity endpoints behind `brreg_roles()`, `brreg_roles_legal()`, `brreg_konsern()`, `brreg_signatur()`, `brreg_prokura()` and `brreg_regnskap()` share one contract. HTTP 404 means the organization number is unknown to that endpoint, which for konsern, signatur, prokura and regnskap is the ordinary case of an entity with nothing registered there: the typed zero-row tibble is returned with attribute `http_status = 404L` and no warning. HTTP 410 means the entity has been deleted: the typed zero-row tibble is returned with `http_status = 410L`, `deletion_date`, and a `tidybrreg_warning_deleted`. Any other status of 400 or above aborts with a `tidybrreg_error_http`. A successful response with nothing registered returns the typed zero-row tibble with no attribute. Use `brreg_entity()` to distinguish a non-existent organization number from a live entity with nothing registered.

See Also

`brreg_signatur()` for general signing authority.

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

Examples

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

---

brreg_regnskap *Retrieve annual accounts (regnskap) from Regnskapsregisteret*

Description

Fetch the key figures of the annual accounts filed by a Norwegian legal entity from the Regnskapsregisteret open API (`https://data.brreg.no/regnskapsregisteret/regnskap/{orgnr}`, OpenAPI 1.1.16). One row per filing is returned. Since 2026-10-01 the default response carries up to three fiscal years per account type, consolidated (`KONSERN`) accounts before company (`SELSKAP`) accounts; requesting an explicit `year` or `account_type` reveals no further years, and years the API no longer serves answer 404.

Usage

```
brreg_regnskap(
  org_nr,
  year = NULL,
  account_type = NULL,
  type = c("code", "label")
)
```

Arguments

org_nr	Character. A single 9-digit organization number, validated with brreg_validate().
year	Integer. A single calendar year of the fiscal period's end date (period_end), including non-calendar fiscal years. NULL (default) returns every year the API serves.
account_type	Character. "SELSKAP" (the company's own accounts) or "KONSERN" (the group accounts). NULL (default) returns both.
type	A type of variables: "code" (default) returns the item, group and statement codes of regnskap_dict and regnskap_groups and the legal form code, "label" returns English labels for them from regnskap_items, regnskap_groups and legal_forms, following the eurostat package's pattern as brreg_entity() does. Pipe to brreg_label() for Norwegian labels or to keep the codes alongside.

Value

A tibble with one row per filing. Always present when a filing is returned: org_nr, filing_id (integer), journal_nr, account_type, period_start and period_end (Date), currency, items (list of tibbles with columns statement, group, item, path, level, is_sum, value), consistent (logical) and inconsistencies (list of character vectors). Frequently present: legal_form, parent_company, liquidation_accounts, presentation_plan, not_audited, audit_waived, small_enterprise, accounting_rules. A zero-row tibble typed on the always-present columns when no filing is available.

Shape

The result is nested in the tidyr sense: the filing is the observational unit of the outer table and its line items are the observational unit of the items table nested in each row. Filing attributes (identity, period, currency, legal form, audit and accounting-principle flags) are scalar columns renamed through the package's regnskap field map (regnskap_dict, published as fields in inst/extdata/endpoints.json). The figures of the three statement objects the API emits (eiendeler, egenkapitalGjeld, resultatregnskapResultat) are rows of items, one per leaf, with:

- statement: assets, equity_liabilities or income_statement.
- group: English name of the object the leaf sits in (current_assets, equity, operating_revenue, ...), from the group map (groups in the endpoint spec).
- item: English name of the leaf (total_assets, cash_and_bank, net_income, ...), from the field map.
- path: the API's dot path, unchanged.

- `level`: depth of the leaf in the API's structure.
- `is_sum`: whether the API names the leaf `sum...`, its own marking of a total.
- `value`: the amount in full units of `currency`.

The API's grouping is structural, not an additive partition: a group's total sits beside components and beside sub-group totals (`sumEiendeler` next to `sumOmloepsmidler` and `sumFordringer`), so adding `value` across items double counts unless `is_sum` and `group` are used. `tidyr::unnest(items)` gives the long form and `tidyr::pivot_wider(names_from = item, values_from = value)` the wide form. As with `brreg_entity()`, only observed fields are emitted and an API field the maps do not know passes through under an auto-generated snake_case name (an unknown leaf inside a statement becomes an item, an unknown object outside the statements becomes scalar columns), so additions BRREG makes to the filing shape are never dropped. An absent leaf is absent, not zero; an object the API sends empty (`{}`) contributes no items.

Consistency flags

Filings are published as filed; the register does not correct them and neither does this function, but it checks the accounting identities the structure implies (balance sheet balances, equity and liabilities add up to their totals, operating and financial results and net income follow from their components; the list is `identities` in the endpoint spec) and flags what fails. `consistent` is `TRUE` when every checkable identity holds within a tolerance of 1 unit of currency, `FALSE` when any fails and `NA` when none is checkable (an identity is checkable when its total and every non-optional term are present). `inconsistencies` is a list-column holding the names of the failing identities, a zero-length character vector when none fails. In the October 2026 population (1,129,601 filings) between 0.1% and 2.5% of filings fail each identity at that tolerance.

Codes are returned by default: `item`, `group` and `statement` are the codes of regnskap_dict and regnskap_groups and `legal_form` the register's code. `type = "label"` or `brreg_label()` translates them from regnskap_items, regnskap_groups and legal_forms in English or Norwegian, inside the nested `items` as well as on the filing row.

The logical account key is `org_nr`, `account_type` and the year of `period_end`; `filing_id` identifies the filing itself (a new id under the same key is a resubmission) and `journal_nr` can be shared by the `SELSKAP` and `KONSERN` filings of one submission.

Unsupported accounting plans

Banks, insurers, pension funds and other entities whose accounting plan (`BANK`, `SKADE`, `LIV`, `PENSJ`, `IDEELL`, ...) the API does not serve answer HTTP 500 with the message `oppstillingsplan som ikke er støttet (PLAN)`. That is a property of the entity, not a transient failure: the typed zero-row tibble is returned with attributes `http_status = 500L` and `unsupported_plan` (the plan code), and a `tidybrreg_warning_unsupported_plan` is signalled. Any other HTTP 500 aborts as documented below.

Status handling

The per-entity endpoints behind `brreg_roles()`, `brreg_roles_legal()`, `brreg_konsern()`, `brreg_signatur()`, `brreg_prokura()` and `brreg_regnskap()` share one contract. HTTP 404 means the organization number is unknown to that endpoint, which for konsern, signatur, prokura and regnskap is the ordinary case of an entity with nothing registered there: the typed zero-row tibble is returned with attribute `http_status = 404L` and no warning. HTTP 410 means the entity has been deleted: the

typed zero-row tibble is returned with `http_status = 410L`, `deletion_date`, and a `tidybrreg_warning_deleted`. Any other status of 400 or above aborts with a `tidybrreg_error_http`. A successful response with nothing registered returns the typed zero-row tibble with no attribute. Use `brreg_entity()` to distinguish a non-existent organization number from a live entity with nothing registered.

See Also

`brreg_entity()` for `last_annual_accounts`, the year of the latest filing the register knows of, `brreg_konsern()` for the group whose KONSERN accounts consolidate an entity, tidybrreg_conditions for the conditions signalled.

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

Examples

```
acc <- brreg_regnskap("923609016")                  # Equinor ASA, all years
acc[c("account_type", "period_end", "consistent")]
tidyr::unnest(acc, items)                            # one row per line item
brreg_regnskap("923609016", account_type = "SELSKAP", year = 2025)

# English labels for items, groups, statements and the legal form
tidyr::unnest(brreg_regnskap("923609016", type = "label"), items)

# Or pipe to brreg_label() for Norwegian labels
brreg_regnskap("923609016") |> brreg_label(lang = "no")
```

---

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. A single 9-digit organization number, validated with `brreg_validate()`.

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_code`, `role_group`, `role_code`, `role`, `first_name`, `middle_name`, `last_name`, `birth_date` (Date), `deceased`, `entity_org_nr`, `entity_name`, `deregistered`, `ordering`, `elected_by`, `group_modified` (Date), `person_id`. A `resigned` column is present only when the payload carries the `fratraadt` field, which BRREG removed on 2026-06-16; live responses no longer include the column, while replays of older data retain it. An entity with no registered roles yields a zero-row tibble with the same typed columns.

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.

Status handling

The per-entity endpoints behind `brreg_roles()`, `brreg_roles_legal()`, `brreg_konsern()`, `brreg_signatur()`, `brreg_prokura()` and `brreg_regnskap()` share one contract. HTTP 404 means the organization number is unknown to that endpoint, which for konsern, signatur, prokura and regnskap is the ordinary case of an entity with nothing registered there: the typed zero-row tibble is returned with attribute `http_status = 404L` and no warning. HTTP 410 means the entity has been deleted: the typed zero-row tibble is returned with `http_status = 410L`, `deletion_date`, and a `tidybrreg_warning_deleted`. Any other status of 400 or above aborts with a `tidybrreg_error_http`. A successful response with nothing registered returns the typed zero-row tibble with no attribute. Use `brreg_entity()` to distinguish a non-existent organization number from a live entity with nothing registered.

See Also

role_types and role_groups for the English lookup tables, tidybrreg_conditions for the conditions signalled.

Other tidybrreg entity functions: `brreg_children()`, `brreg_download()`, `brreg_entity()`, `brreg_konsern()`, `brreg_prokura()`, `brreg_regnskap()`, `brreg_roles_legal()`, `brreg_search()`, `brreg_signatur()`, `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. A single 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), deregistered. A resigned column is present only when the payload carries the fratraadt field, removed by BRREG on 2026-06-16. An entity holding no roles elsewhere yields a zero-row tibble with the same typed columns.

Status handling

The per-entity endpoints behind brreg_roles(), brreg_roles_legal(), brreg_konsern(), brreg_signatur(), brreg_prokura() and brreg_regnskap() share one contract. HTTP 404 means the organization number is unknown to that endpoint, which for konsern, signatur, prokura and regnskap is the ordinary case of an entity with nothing registered there: the typed zero-row tibble is returned with attribute http_status = 404L and no warning. HTTP 410 means the entity has been deleted: the typed zero-row tibble is returned with http_status = 410L, deletion_date, and a tidybrreg_warning_deleted. Any other status of 400 or above aborts with a tidybrreg_error_http. A successful response with nothing registered returns the typed zero-row tibble with no attribute. Use brreg_entity() to distinguish a non-existent organization number from a live entity with nothing registered.

See Also

brreg_roles() for who holds roles in an entity.

Other tidybrreg entity functions: brreg_children(), brreg_download(), brreg_entity(), brreg_konsern(), brreg_prokura(), brreg_regnskap(), brreg_roles(), brreg_search(), brreg_signatur(), 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.

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. For privacy the register refuses a range that lies within one to four employees (HTTP 400, tidybrreg_error_http with the register's message); ranges starting at 0 or at 5 and above are served.
bankrupt	Logical. If TRUE, return only bankrupt entities. Only supported for registry = "enheter"; supplying it for "underenheter" is an error.
parent_org_nr	Character. Filter to subsidiaries of this org.
max_results	Integer. Maximum entities to return (default 200). Values above the 10,000 result window are allowed but abort when the register reports more matches than the window can deliver.
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); only observed fields are emitted (see brreg_entity()). Attributes: total_matches, the number of matches the register reported, and complete, FALSE when fewer rows were returned than matched. A valid empty result is a zero-row tibble with character columns org_nr, name and legal_form.

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.

Result window

The search endpoints allow a page size of up to 10,000 but reject any page where `size * (page + 1)` exceeds 10,000 (observed 2026-08-08: a Bergen municipality query reported 57,954 matches; result 10,001 answered HTTP 400 `size * (page+1) kan ikke overstige 10_000`). A query matching more entities than that window cannot be retrieved completely through search. When the register reports more matches than the window and `max_results` asks for more than the window, the call aborts with a `tidybrreg_error_result_window` before any page is returned; narrow the query (for example by `municipality_code`, `legal_form` or `nace_code`), or use `brreg_download()` for the full register. When `max_results` is at or below the window the first `max_results` matches are returned and the result carries `complete = FALSE` plus a `tidybrreg_warning_truncated` whenever more matches exist.

Tidy shape

Every tibble the package returns is tidy in Wickham's sense: one observation per row, one variable per column, one observational unit per table. The register nests two kinds of array inside an entity. Arrays of strings (`business_address`, `postal_address`, `location_address`, `purpose`, `activity`) are the line fragments of one value and are joined with `"\n"`, which no fragment contains, so `strsplit(x, "\n")` recovers the lines. Arrays of objects (`historiske_navn`, `annotations`, any array of objects the register adds) are observations of another unit and are nested as a list-column of tibbles, one row per element with snake_case columns, `NULL` for an entity without them; `tidyr::unnest()` expands them to their own table. An empty array is not an observation and emits no column. Nothing is serialized to JSON text.

See Also

`brreg_entity()` for single lookups, `brreg_label()` for translating codes to English, tidybrreg_conditions for the errors and warnings signalled.

Other tidybrreg entity functions: `brreg_children()`, `brreg_download()`, `brreg_entity()`, `brreg_konsern()`, `brreg_prokura()`, `brreg_regnskap()`, `brreg_roles()`, `brreg_roles_legal()`, `brreg_signatur()`, `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_signatur` *Retrieve signature authority (signaturrett)*

Description

Fetch the registered signing combinations for a Norwegian legal entity from the Brønnøysund Fullmakt service. Signature authority determines who may bind the entity in general, and is distinct from the roles returned by `brreg_roles()`: an entity may have many board members yet a signing rule such as "the board jointly" or "the general manager alone".

Usage

```
brreg_signatur(org_nr)
```

Arguments

`org_nr`	Character. A single 9-digit organization number.

Details

Each registered combination (`combination_id`) carries a rule (`rule`, e.g. "Styret i fellesskap") and the persons who satisfy it. One row is returned per person within each combination.

Value

A tibble with one row per person per signing combination. Columns: `org_nr`, `entity_name`, `signature_type` (`"signature"`), `rule_status`, `rule_text`, `combination_id`, `combination_code`, `rule`, `name`, `birth_date` (Date), `role_code`, `role`. A zero-row tibble with the same typed columns if the entity has no registered signing combination.

Person identification

The Fullmakt service returns each person's name as a single string in `name`; it does not split it into given and family names, so no synthetic `person_id` is constructed. Join to `brreg_roles()` on `birth_date` together with `name` at query time if cross-referencing to the role network is required. Role designations are returned as English labels looked up from role_types, with the original Norwegian code preserved in `role_code`; the signing-mode designations (`SIGN`, `SIFE`, `SIHV`, `PROK`, `POFE`, `POHV`) are included in role_types alongside the board roles. Birth dates arrive as `DD.MM.YYYY` from this service and as ISO dates from the roles endpoint; both are parsed to `Date`, so the two join directly.

Standardised vs registered rule

`rule` and `combination_code` come from the registry's structured combination. Where the rule has been standardised (`rule_status` code `"RF"`) this is the registered rule. Where it has not (`"RI"`), the structured combination is the statutory default ("the board jointly") and the actually registered rule is the free text in `rule_text`; read `rule_text` in that case. `rule_text` may be `NA` when no free text is recorded.

Rules the service declines to resolve

For some entities (Equinor ASA among them) the service answers HTTP 200 with no combinations and a routine status instead: code `NA`, "Generelle unntak, finnes roller som har status det ikke kan foretas maskinelt oppslag mot", meaning a rule is registered but cannot be resolved mechanically. The typed zero-row tibble is then returned with attributes `http_status = 200L`, `routine_status` and `routine_text`, and a `tidybrreg_warning_unresolved` is signalled, so the case is distinguishable from an entity with nothing registered (no attribute, no warning).

Status handling

The per-entity endpoints behind `brreg_roles()`, `brreg_roles_legal()`, `brreg_konsern()`, `brreg_signatur()`, `brreg_prokura()` and `brreg_regnskap()` share one contract. HTTP 404 means the organization number is unknown to that endpoint, which for konsern, signatur, prokura and regnskap is the ordinary case of an entity with nothing registered there: the typed zero-row tibble is returned with attribute `http_status = 404L` and no warning. HTTP 410 means the entity has been deleted: the typed zero-row tibble is returned with `http_status = 410L`, `deletion_date`, and a `tidybrreg_warning_deleted`. Any other status of 400 or above aborts with a `tidybrreg_error_http`. A successful response with nothing registered returns the typed zero-row tibble with no attribute. Use `brreg_entity()` to distinguish a non-existent organization number from a live entity with nothing registered.

See Also

`brreg_prokura()` for procuration, `brreg_roles()` for the underlying role assignments.

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

Examples

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

---

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). When the parent has more, the result carries complete = FALSE and a tidybrreg_warning_truncated is signalled.
type	One of "code" (default) or "label".

Value

A tibble with one row per sub-unit. Same column schema and attributes 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_children(), brreg_download(), brreg_entity(), brreg_konsern(), brreg_prokura(), brreg_regnskap(), brreg_roles(), brreg_roles_legal(), brreg_search(), brreg_signatur(), 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, path = 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, POSIXct, or ISO 8601 string. Return updates at or after this instant. Defaults to yesterday (midnight UTC).
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" or "underenheter".
verbose	Logical. Print page-level progress.

Details

Patches are RFC 6902 operations. Nested values are flattened to one row per leaf; path keeps the raw JSON pointer of the leaf (for example /forretningsadresse/adresse/0) and field its snake_case rendering, so the register's own representation is never lost. A move emits an add at the destination and a remove at the source.

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 (POSIXct, UTC), operation, field, path, new_value. No list-columns. Events without field patches appear as rows with operation, field, path and new_value all NA.

Time handling

since is sent to the feed as a UTC instant. A POSIXct keeps its full precision; a Date is midnight UTC of that day; an ISO 8601 string ("2026-03-01" or "2026-03-01T14:00:00Z") is parsed as UTC. Timestamps in the result are POSIXct in the UTC time zone, whatever the session time zone.

Completeness

The feed is read page by page until a page is short, the feed is exhausted, or max_pages is reached. The result carries attributes complete (FALSE when max_pages stopped the read while the last page was full, or a later page failed), pages_fetched, and cursor (the last update_id read; pass update_id + 1 as the next cursor to resume). Stopping on max_pages signals a tidybrreg_warning_truncated; a failure after the first page signals a tidybrreg_warning_partial and returns the pages already read; a failure on the first page aborts with a tidybrreg_error_http.

Unknown members

Any member of an update object that the package does not model is passed through as a column named by its snake_case key, and the result carries attribute unknown_members plus a single tidybrreg_warning_unknown_member, so an upstream addition cannot vanish silently.

See Also

brreg_updates() for the event-level view.

Other tidybrreg entity functions: brreg_children(), brreg_download(), brreg_entity(), brreg_konsern(), brreg_prokura(), brreg_regnskap(), brreg_roles(), brreg_roles_legal(), brreg_search(), brreg_signatur(), 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 instant. 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, POSIXct, or ISO 8601 string. Return updates at or after this instant. Defaults to yesterday (midnight UTC).
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. Not available for type = "roller", whose feed carries no field patches; supplying TRUE there is an error.
type	One of "enheter", "underenheter", or "roller". The roller feed uses the CloudEvents format and reports only which entity's roles changed; fetch the new state with brreg_roles().
verbose	Logical. Print page-level progress.

Value

A tibble with columns: `update_id` (integer), `org_nr`, `change_type`, `timestamp` (POSIXct, UTC). If `include_changes = TRUE`, a list-column `changes` with tibbles of `operation`, `field`, `path`, `new_value`. An empty feed yields a zero-row tibble with the same typed columns.

Time handling

`since` is sent to the feed as a UTC instant. A `POSIXct` keeps its full precision; a `Date` is midnight UTC of that day; an ISO 8601 string (`"2026-03-01"` or `"2026-03-01T14:00:00Z"`) is parsed as UTC. Timestamps in the result are `POSIXct` in the `UTC` time zone, whatever the session time zone.

Completeness

The feed is read page by page until a page is short, the feed is exhausted, or `max_pages` is reached. The result carries attributes `complete` (`FALSE` when `max_pages` stopped the read while the last page was full, or a later page failed), `pages_fetched`, and `cursor` (the last `update_id` read; pass `update_id + 1` as the next cursor to resume). Stopping on `max_pages` signals a `tidybrreg_warning_truncated`; a failure after the first page signals a `tidybrreg_warning_partial` and returns the pages already read; a failure on the first page aborts with a `tidybrreg_error_http`.

Unknown members

Any member of an update object that the package does not model is passed through as a column named by its snake_case key, and the result carries attribute `unknown_members` plus a single `tidybrreg_warning_unknown_member`, so an upstream addition cannot vanish silently.

See Also

`brreg_update_fields()` for a flat alternative, tidybrreg_conditions for the conditions signalled.

Other tidybrreg entity functions: `brreg_children()`, `brreg_download()`, `brreg_entity()`, `brreg_konsern()`, `brreg_prokura()`, `brreg_regnskap()`, `brreg_roles()`, `brreg_roles_legal()`, `brreg_search()`, `brreg_signatur()`, `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 (the series the register allocates from), 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 (including `NA` input).

See Also

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

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

Examples

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

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, on the single-entity lookups, the searches and the JSON bulk alike. 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 101 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"`.

Source

Package-authored; see tidybrreg_provenance.

See Also

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

Other tidybrreg reference data: `legal_forms`, `regnskap_dict`, `regnskap_groups`, `regnskap_items`, `role_groups`, `role_types`, `tidybrreg_provenance`

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 (SSB classification 6, SN2007) or institutional sector codes (SSB classification 39). Dictionaries are cached in a session-level environment (following the eurostat package pattern), so the network is consulted at most once per dictionary and language per session.

Usage

```
get_brreg_dic(
  dictname = c("nace", "sector"),
  lang = c("en", "no"),
  source = c("auto", "klass", "bundled"),
  refresh = FALSE
)
```

Arguments

`dictname`	One of `"nace"` or `"sector"`.
`lang`	`"en"` (default) for English labels or `"no"` for Norwegian.
`source`	One of `"auto"`, `"klass"`, or `"bundled"`; see the section on network access.
`refresh`	Logical. If `TRUE`, ignore the session cache and fetch again.

Value

A tibble with columns `code`, `name` (label in the requested language), and `level`. Attributes `source` (`"klass"` or `"bundled"`), `lang`, and `vintage` (the date the labels were retrieved from KLASS) record where the labels came from.

Network access and offline use

With `source = "auto"` (default) the dictionary is requested from the SSB KLASS API (`data.ssb.no`), which is the only source of Norwegian labels and of codes valid today. If the request fails, the bundled English dictionary is returned instead and a `tidybrreg_warning_bundled_fallback` states its vintage; for `lang = "no"` there is no bundled fallback and the error propagates. `source = "bundled"` never touches the network and is the mode to use offline or in tests; `options(tidybrreg.offline = TRUE)` makes it the default for the session. `source = "klass"` requires the network and errors when it is unavailable. The bundled dictionaries were retrieved from KLASS on 2026-06-16 (see tidybrreg_provenance).

See Also

`brreg_label()` for translating columns in place, tidybrreg_provenance for the bundled dictionaries' sources.

Other tidybrreg utilities: `brreg_label()`, `brreg_validate()`, `tidybrreg_conditions`, `tidybrreg_endpoints`

Examples

```
get_brreg_dic("nace", source = "bundled")
attr(get_brreg_dic("sector", source = "bundled"), "vintage")

get_brreg_dic("nace", lang = "no")
```

---

legal_forms *Norwegian legal form codes with English translations*

Description

All organisasjonsformer registered with the Brønnøysund Register Centre, with English translations. Codes and Norwegian names are fetched from the brreg API during package build; the English names are manual translations by the package author.

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.

Source

Brønnøysund Register Centre, https://data.brreg.no/enhetsregisteret/api/organisasjonsformer, NLOD 2.0, retrieved 2026-06-16; see tidybrreg_provenance.

See Also

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

Other tidybrreg reference data: field_dict, regnskap_dict, regnskap_groups, regnskap_items, role_groups, role_types, tidybrreg_provenance

Examples

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

---

regnskap_dict *Regnskap field dictionary: filing JSON paths to English column names*

Description

A tibble mapping dot paths of the Regnskapsregisteret filing JSON to English column names and R types, with the columns of field_dict. brreg_regnskap() applies it exactly as brreg_entity() applies field_dict: leaves outside the three statement objects become the filing's scalar columns, leaves inside them become the item codes of the nested items table, and paths absent from the dictionary pass through with auto-generated snake_case names. Source spellings such as regnkapsprinsipper and sumInnskuttEgenkaptial are the API's own.

Usage

```
regnskap_dict
```

Format

A tibble with 46 rows and 3 columns:

api_path Dot-notation path in the filing JSON (e.g. "regnskapsperiode.tilDato", "eiendeler.sumEiendeler").

col_name English name used in package output: a column of the filing row or an item code (e.g. "period_end", "total_assets").

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

Source

Package-authored; see tidybrreg_provenance.

See Also

brreg_regnskap(), regnskap_groups, regnskap_items.

Other tidybrreg reference data: field_dict, legal_forms, regnskap_groups, regnskap_items, role_groups, role_types, tidybrreg_provenance

Examples

```
regnskap_dict
regnskap_dict[regnskap_dict$type == "numeric", ]
```

---

regnskap_groups *Regnskap groups: the objects that group line items, with labels*

Description

One row per object of the filing JSON that contains line items, from the three statements (eiendeler, egenkapitalGjeld, resultatregnskapResultat, level 0) down to the leaf groups. The code is the group (and, for level-0 rows, the statement) of the items table of brreg_regnskap(); brreg_label() translates it to name_en or name_no, as role_groups does for role groups.

Usage

```
regnskap_groups
```

Format

A tibble with 17 rows and 4 columns:

path Dot-notation path of the object in the filing JSON.

code Group code used in package output (e.g. "assets", "current_liabilities").

name_en English name.

name_no Norwegian name, the presentation-plan caption.

Source

Package-authored; see tidybrreg_provenance.

See Also

brreg_regnskap(), regnskap_dict, regnskap_items.

Other tidybrreg reference data: field_dict, legal_forms, regnskap_dict, regnskap_items, role_groups, role_types, tidybrreg_provenance

Examples

```
regnskap_groups
brreg_label(c("assets", "operating"), dic = "regnskap_group", lang = "no")
```

`regnskap_items` *Regnskap line items with English and Norwegian labels*

Description

One row per line item of the filing, keyed by the `item` code that regnskap_dict assigns to each statement leaf. `brreg_label()` translates the `item` column of `brreg_regnskap()`'s `items` table to `name_en` or `name_no`, as role_types does for role codes.

Usage

```
regnskap_items
```

Format

A tibble with 31 rows and 3 columns:

code Item code used in package output (e.g. `"total_assets"`, `"net_income"`).

name_en English label.

name_no Norwegian label, the presentation-plan caption.

Source

Package-authored; see tidybrreg_provenance.

See Also

`brreg_regnskap()`, regnskap_dict, regnskap_groups.

Other tidybrreg reference data: `field_dict`, `legal_forms`, `regnskap_dict`, `regnskap_groups`, `role_groups`, `role_types`, `tidybrreg_provenance`

Examples

```
regnskap_items
brreg_label(c("total_assets", "net_income"), dic = "regnskap_item", lang = "no")
```

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.

Source

Package-authored from BRREG documentation; see tidybrreg_provenance.

See Also

`brreg_roles()`, role_types.

Other tidybrreg reference data: `field_dict`, `legal_forms`, `regnskap_dict`, `regnskap_groups`, `regnskap_items`, `role_types`, `tidybrreg_provenance`

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 24 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.

Source

Package-authored from BRREG documentation; see tidybrreg_provenance.

See Also

`brreg_roles()`, role_groups.

Other tidybrreg reference data: `field_dict`, `legal_forms`, `regnskap_dict`, `regnskap_groups`, `regnskap_items`, `role_groups`, `tidybrreg_provenance`

Examples

```
role_types
```

---

tidybrreg_conditions *Conditions signalled by tidybrreg*

Description

Every error and warning raised by the package carries a class that can be caught with `tryCatch()` or silenced with `suppressWarnings()` selectively, and a set of fields describing the request that produced it. All classes inherit from `tidybrreg_error` or `tidybrreg_warning`.

Errors

- `tidybrreg_error_invalid_org_nr`: the organization number failed `brreg_validate()`. Field `org_nr`.
- `tidybrreg_error_invalid_filter`: a filter is not supported by the requested registry (for example `bankrupt` on `underenheter`). Fields `registry`, `filter`.
- `tidybrreg_error_not_found`: the endpoint answered HTTP 404 for an organization number. Fields `org_nr`, `endpoint`, `status`.
- `tidybrreg_error_http`: any other HTTP status of 400 or above, or a failure on the first page of a paginated request. Fields `endpoint`, `status`, `messages` (as extracted from the BRREG error body). 429, 503, 500, 502 and 504 are retried up to three times first (except the Regnskapsregisteret 500 for an unsupported accounting plan, which is final).
- `tidybrreg_error_rate_limited`: the register answered its rate limit as HTTP 200 with the text `Too many requests` on every retry. Fields `url`, `status` (200), `attempts`.

- `tidybrreg_error_connection`: the host could not be reached (DNS, TCP, TLS or a dropped connection) on every retry. Fields `url`, `attempts`.
- `tidybrreg_error_result_window`: the query matches more entities than BRREG's search window can deliver and the request asked for more than the window. Fields `endpoint`, `total_matches`, `retrieval_limit`, `filters`. See `brreg_search()`.

Warnings

- `tidybrreg_warning_deleted`: the entity has been deleted from the register (HTTP 410, or HTTP 200 with `respons_klasse "SlettetEnhet"`). Fields `org_nr`, `deletion_date`.
- `tidybrreg_warning_partial`: a paginated request failed after at least one page was retrieved; the partial result is returned with attribute `complete = FALSE`. Fields `endpoint`, `status`, `pages_fetched`.
- `tidybrreg_warning_truncated`: more results exist than were returned because of `max_results` or `max_pages`; the result carries attribute `complete = FALSE`. Fields `endpoint`, `returned`, `total_matches`.
- `tidybrreg_warning_unknown_member`: an update feed carried a member the package does not model; it is passed through unchanged. Field `members`.
- `tidybrreg_warning_bundled_fallback`: SSB KLASS was unreachable and a bundled dictionary was used instead. Fields `dictname`, `vintage`.
- `tidybrreg_warning_unresolved`: the Fullmakt service answered HTTP 200 but declined to resolve the signature or procuration rule mechanically (`status.rutineStatus`); the typed empty result carries `http_status = 200L`, `routine_status` and `routine_text`. Fields `org_nr`, `signature_type`, `routine_status`, `routine_text`. See `brreg_signatur()`.
- `tidybrreg_warning_unsupported_plan`: Regnskapsregisteret does not serve the entity's accounting plan (banks, insurers, pension funds); the typed empty result carries `http_status = 500L` and `unsupported_plan`. Fields `org_nr`, `plan`, `api_message`. See `brreg_regnskap()`.
- `tidybrreg_warning_parse`: values could not be coerced to the type declared in field_dict; the result carries attribute `brreg_parse_problems`. Field `n`.

See Also

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

Examples

```
tryCatch(
  brreg_entity("123456789"),
  tidybrreg_error_invalid_org_nr = function(e) e$org_nr
)
```

`tidybrreg_endpoints` *Retrieval limits of the register endpoints*

Description

One row per endpoint reachable through the fifteen exports, with the pagination model, the retrieval window, and how the package behaves at the boundary. The evidence column separates what BRREG documents (https://data.brreg.no/enhetsregisteret/api/dokumentasjon/) from what has been observed against the live service; limits observed for one route are never assumed for another. Boundary probes for the search routes run in the scheduled live test workflow (`tests/testthat/test-liveapi-audit.R`).

Enhetsregisteret API

Base URL `https://data.brreg.no/enhetsregisteret/api`.

Route	Function
`GET /enheter/{orgnr}`	`brreg_entity()`
`GET /underenheter/{orgnr}`	`brreg_entity()`
`GET /enheter?...`	`brreg_search()`, `brreg_children()`
`GET /underenheter?...`	`brreg_search()`, `brreg_underenheter()`
`GET /enheter/{orgnr}/roller`	`brreg_roles()`
`GET /roller/enheter/{orgnr}/juridiskeroller`	`brreg_roles_legal()`
`GET /konsernstruktur/{orgnr}`	`brreg_konsern()`
`GET /oppdateringer/enheter`, `/oppdateringer/underenheter`	`brreg_updates()`, `brreg_update_fields(`
`GET /oppdateringer/roller`	`brreg_updates()` with `type = "roller"`
`GET /enheter/lastned`, `/underenheter/lastned`, `/roller/totalbestand`	`brreg_download()`
`GET /organisasjonsformer`	build script only

Fullmakt service

Base URL `https://data.brreg.no/fullmakt`.

Route	Function	Pagination	Limits and boundary
`GET /enheter/{orgnr}/signatur`	`brreg_signatur()`	none	404 when nothing is registered (common for ENK)
`GET /enheter/{orgnr}/prokura`	`brreg_prokura()`	none	as signatur

Regnskapsregisteret API

Base URL `https://data.brreg.no/regnskapsregisteret/regnskap` (OpenAPI 1.1.16).

Route	Function	Pagination	Limits and boundary
`GET /{orgnr}`, optional `år`, `regnskapstype`	`brreg_regnskap()`	none (whole array)	up to three fiscal years per type si

SSB KLASS (not BRREG)

Base URL `https://data.ssb.no/api/klass/v1`.

Route	Function	Pagination	Limits and boun
`GET /classifications/6/codesAt`, `/classifications/39/codesAt`	`get_brreg_dic()`	none	whole code list

Rate limiting

BRREG publishes no rate limit or rate-limit headers; the package throttles itself to 5 requests per second per host and retries 429, 503, 500, 502 and 504 up to three times, as well as connection failures and the register's rate limit when it arrives as HTTP 200 with the text `Too many requests` (then a `tidybrreg_error_rate_limited`). The Regnskapsregisteret 500 for an unsupported accounting plan is final. Request timeouts are httr2's defaults.

See Also

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

`tidybrreg_provenance` *Provenance of the bundled reference data*

Description

Every table shipped with tidybrreg is listed here with its source, licence, vintage, the transformations applied, and how the package uses it. The tables are regenerated by `data-raw/build_dictionaries.R`; the vintages below are the dates of the last regeneration that touched each table, taken from the repository history.

Exported datasets (`data/`)

field_dict **Package-authored.** A mapping from BRREG JSON paths to English column names and R types, written by the package author from the Enhetsregisteret API documentation (https://data.brreg.no/enhetsregisteret/api/dokumentasjon/) and from fields observed on live responses. The Norwegian paths are BRREG's identifiers; the English names and types are the author's editorial choices and carry the package's MIT licence. Last revised 2026-06-16. Used at runtime by every parser.

legal_forms **BRREG code list plus manual translation.** `code`, `name_no` and `expired` are retrieved from `GET https://data.brreg.no/enhetsregisteret/api/organisasjonsformer`, published by the Brønnøysund Register Centre under the Norwegian Licence for Open Government Data (NLOD) 2.0 (https://data.norge.no/nlod/en/2.0); `name_en` is a manual translation by the package author (unreviewed by BRREG), falling back to `name_no` for codes without a translation. Retrieved 2026-06-16. Used by `brreg_label()`.

role_types **Package-authored from BRREG documentation.** Codes and Norwegian descriptions follow the role codes listed in the Enhetsregisteret roller documentation and the Fullmakt service; English names are a manual translation by the package author. BRREG publishes no machine-readable role code list with translations. Last revised 2026-06-16. Used by `brreg_roles()`, `brreg_roles_legal()`, `brreg_signatur()`, `brreg_prokura()` and `brreg_label()`.

role_groups **Package-authored from BRREG documentation.** As role_types, for role group (`rollegruppe`) codes. Last revised 2026-06-16. Used by `brreg_roles()` and `brreg_label()`.

regnskap_dict **Package-authored.** As field_dict, for the Regnskapsregisteret filing: dot paths of the filing JSON (OpenAPI 1.1.16, `https://data.brreg.no/regnskapsregisteret/regnskap/v3/api-docs`, population-verified against the brreg-json-fetcher `data-2026-10` release) to English names and R types. Written 2026-10-30. Used by `brreg_regnskap()`.

regnskap_groups **Package-authored.** The objects that group line items in the filing, with English and Norwegian names; the level-0 rows are the three statements. The Norwegian names are the presentation-plan captions of the Norwegian Accounting Act; English is a manual translation by the package author. Written 2026-10-30. Used by `brreg_regnskap()` and `brreg_label()`.

regnskap_items **Package-authored.** English and Norwegian labels of the line items named in regnskap_dict, as role_types is for role codes. Written 2026-10-30. Used by `brreg_label()`.

Internal fallback dictionaries (`R/sysdata.rda`)

`nace_codes` **Statistics Norway (SSB) KLASS classification 6** (Standard Industrial Classification SN2007), English labels, retrieved from `https://data.ssb.no/api/klass/v1/classifications/6/codesAt` on 2026-06-16 (the effective date of the codes is that retrieval date). SSB publishes KLASS classifications as open data under NLOD 2.0 according to its data licence page (`https://www.ssb.no/en/omssb/tjenester-og-verktoy/api/klass-api`); the maintainer has not obtained a separate confirmation, so the statement is reproduced as published. Used only when SSB KLASS is unreachable or `source = "bundled"` is requested in `get_brreg_dic()`; the returned tibble carries `vintage`.

`sector_codes` **SSB KLASS classification 39** (institutional sector), English labels, same source, licence statement, retrieval date and use as `nace_codes`.

Register data

Entity, role, group, signature and update data retrieved at runtime come from the Brønnøysund Register Centre's open APIs and are licensed under NLOD 2.0; the licence requires attribution to the Brønnøysund Register Centre when the data are redistributed. KLASS labels retrieved at runtime are Statistics Norway's.

See Also

Other tidybrreg reference data: `field_dict`, `legal_forms`, `regnskap_dict`, `regnskap_groups`, `regnskap_items`, `role_groups`, `role_types`

Index

∗ **datasets**
field_dict, 26
legal_forms, 28
regnskap_dict, 29
regnskap_groups, 30
regnskap_items, 31
role_groups, 32
role_types, 32
∗ **tidybrreg entity functions**
brreg_children, 3
brreg_download, 4
brreg_entity, 6
brreg_konsern, 8
brreg_prokura, 11
brreg_regnskap, 12
brreg_roles, 15
brreg_roles_legal, 16
brreg_search, 17
brreg_signatur, 20
brreg_underenheter, 21
brreg_update_fields, 22
brreg_updates, 24
∗ **tidybrreg reference data**
field_dict, 26
legal_forms, 28
regnskap_dict, 29
regnskap_groups, 30
regnskap_items, 31
role_groups, 32
role_types, 32
tidybrreg_provenance, 36
∗ **tidybrreg utilities**
brreg_label, 9
brreg_validate, 26
get_brreg_dic, 27
tidybrreg_conditions, 33
tidybrreg_endpoints, 35

brreg_children, 3, *6*, *8, 9*, *12*, *15–17*, *19*, *21, 22*, *24, 25*
brreg_children(), *8, 9*, *22*, *35*
brreg_download, *3*, 4, *8, 9*, *12*, *15–17*, *19*, *21, 22*, *24, 25*
brreg_download(), *19*, *35*
brreg_entity, *3*, 6, *6*, *9*, *12*, *15–17*, *19*, *21, 22*, *24, 25*
brreg_entity(), *3*, *5*, *9, 10*, *12–19*, *21–23*, *26, 27*, *29*, *35*
brreg_konsern, *3*, *6*, 8, *8*, *12*, *15–17*, *19*, *21, 22*, *24, 25*
brreg_konsern(), *3*, *9*, *12*, *14–17*, *21*, *35*
brreg_label, 9, *26*, *28*, *34*, *36*
brreg_label(), *7, 8*, *13, 14*, *19*, *28–32*, *36, 37*
brreg_prokura, *3*, *6*, *8, 9*, 11, *15–17*, *19*, *21, 22*, *24, 25*
brreg_prokura(), *9*, *12*, *14*, *16, 17*, *21*, *35, 36*
brreg_regnskap, *3*, *6*, *8, 9*, 12, *12*, *16, 17*, *19*, *21, 22*, *24, 25*
brreg_regnskap(), *9, 10*, *12*, *14*, *16, 17*, *21*, *29–31*, *34, 35*, *37*
brreg_roles, *3*, *6*, *8, 9*, *12*, 15, *15*, *17*, *19*, *21, 22*, *24, 25*
brreg_roles(), *4*, *9–12*, *14*, *16, 17*, *20, 21*, *24*, *32, 33*, *35–37*
brreg_roles_legal, *3*, *6*, *8, 9*, *12*, *15*, 16, *16*, *19*, *21, 22*, *24, 25*
brreg_roles_legal(), *9*, *12*, *14*, *16, 17*, *21*, *35, 36*
brreg_search, *3*, *6*, *8, 9*, *12*, *15, 16*, 17, *17*, *21, 22*, *24, 25*
brreg_search(), *3–6*, *8*, *10*, *21, 22*, *34, 35*
brreg_signatur, *3*, *6*, *8, 9*, *12*, *15–17*, *19*, 20, *22*, *24, 25*
brreg_signatur(), *9*, *11, 12*, *14*, *16, 17*, *21*, *34–36*
brreg_underenheter, *3*, *6*, *8, 9*, *12*, *15–17*, *19*, 21, *21*, *24, 25*
brreg_underenheter(), *3*, *35*
brreg_update_fields, *3*, *6*, *8, 9*, *12*, *15–17*,

19, *21*, 22, *22*, *25*
brreg_update_fields(), *25*, *35*
brreg_updates, *3*, *6*, *8*, *9*, *12*, *15–17*, *19*, *21*, *22*, 24, *24*
brreg_updates(), *6*, *24*, *35*
brreg_validate, *10*, 26, *28*, *34*, *36*
brreg_validate(), *6*, *13*, *15*, *33*

field_dict, *4*, *5*, *7*, *8*, *18*, 26, *29–34*, *36*, *37*

get_brreg_dic, *10*, *26*, 27, *34*, *36*
get_brreg_dic(), *7*, *10*, *36*, *37*

lapply(), *6*
legal_forms, *10*, *13*, *14*, *18*, *19*, *27*, 28, *30–33*, *36*, *37*

regnskap_dict, *13*, *14*, *27*, 29, *29–33*, *37*
regnskap_groups, *10*, *13*, *14*, *27*, *29*, 30, *30–33*, *37*
regnskap_items, *10*, *13*, *14*, *27*, *29*, *30*, 31, *32*, *33*, *37*
role_groups, *10*, *15*, *16*, *27*, *29–31*, 32, *33*, *37*
role_types, *10*, *11*, *15*, *16*, *20*, *27*, *29–31*, 32, *32*, *36*, *37*

suppressWarnings(), *33*

tidybrreg_conditions, *8*, *10*, *15*, *16*, *19*, *25*, *26*, *28*, 33, *36*
tidybrreg_endpoints, *10*, *26*, *28*, *34*, 35
tidybrreg_provenance, *27–33*, 36
tryCatch(), *33*