

Package: ecospgr (via r-universe)

October 19, 2026

Title Shared Loader Utilities for ecospg

Version 0.5.2

Description Extracts duplicated R code from ecospg data loaders into a reusable package. Provides database connection, catalog registration, partitioned table management, staging/ingest, S3 upload, and CLI argument parsing utilities.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Imports DBI, RPostgres, glue, cli, jsonlite, fs, sf

Suggests testthat (>= 3.0.0), withr

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev make libuv1-dev libssl-dev libpq-dev libproj-dev libsqlite3-dev libudunits2-dev

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

Date/Publication 2026-10-19 09:57:16 UTC

RemoteUrl <https://gitlab.au.dk/ecos/tools/r-pkgs/ecospgr>

RemoteRef HEAD

RemoteSha d6e124c097ad99d46e02b3e1fe7e4e01fa8008fb

Contents

epg_build_source_meta	2
epg_check_feature_count	3
epg_collect_source_meta_overrides	3
epg_create_partition	4
epg_detect_xsd_for_gml	4
epg_dfd_auth_params	5
epg_dfd_auth_qs	5
epg_drop_version	6

epg_ensure_dataset	7
epg_ensure_topic	7
epg_finalize_version	8
epg_fix_encoding	8
epg_insert_normalized	9
epg_parse_arg	10
epg_parse_json_arg	10
epg_parse_json_payload	11
epg_persist_source_meta	11
epg_pg_connect	12
epg_promote_latest	12
epg_read_json_file	13
epg_redact_url	13
epg_register_version	14
epg_resolve_s3_config	15
epg_run_pipeline	15
epg_sanitize_name	17
epg_set_raw_file_url	17
epg_shapefile_encoding_opts	18
epg_upload_raw_files	18
epg_url_without_query	19
epg_write_staging	19

Index 20

epg_build_source_meta *Build base source metadata*

Description

source_url is stored in meta.dataset_version.source_meta, which warehouse_reader can read, so any query string is stripped here — a download URL that carries an API key must not become catalog metadata.

Usage

```
epg_build_source_meta(source_url, metadata_url = NULL, extra = list())
```

Arguments

source_url	The source URL for the download.
metadata_url	Optional metadata URL.
extra	Named list of additional metadata fields to merge.

Value

A named list of source metadata.

epg_check_feature_count

Check a loaded feature count against what the source reported

Description

Upstream services silently cap responses (WFS at count, ArcGIS at maxRecordCount), so a load can succeed while holding only part of the data. Call this with the count the source reported before publishing.

Usage

```
epg_check_feature_count(n_loaded, n_expected, tolerance = 0, what = "dataset")
```

Arguments

n_loaded	Features actually loaded.
n_expected	Features the source reported, or NA if unknown.
tolerance	Fraction of n_expected that may be missing without error. Defaults to 0 — an exact match is required.
what	Label used in messages.

Value

TRUE invisibly when the count is acceptable.

epg_collect_source_meta_overrides

Collect source metadata overrides from file and/or inline JSON

Description

Reads optional JSON from a file path and/or an inline JSON string and merges them (inline takes precedence over file).

Usage

```
epg_collect_source_meta_overrides(json_file_path = NULL, json_inline = NULL)
```

Arguments

json_file_path	Path to a JSON file (or NULL).
json_inline	Inline JSON string or @file reference (or NULL).

Value

A named list of override fields (may be empty).

`epg_create_partition` *Create a partition table and indexes for a dataset version*

Description

Uses `glue_sql()` for safe SQL interpolation. Identifier components are validated to contain only safe characters (alphanumeric + underscore).

Usage

```
epg_create_partition(con, dataset, ver_id, geom_type = "MultiPolygon")
```

Arguments

<code>con</code>	A DBI connection.
<code>dataset</code>	Dataset name (used as part of table names in vector schema).
<code>ver_id</code>	The <code>dataset_version.id</code> (integer).
<code>geom_type</code>	Geometry type constraint for the partition. Defaults to "MultiPolygon". Use "Geometry" for broader geometry support (e.g., geodk datasets).

`epg_detect_xsd_for_gml`

Detect a companion XSD file for a GML file

Description

Looks for `schemaLocation` hints in the GML header, then falls back to filename-based matching and sibling XSD files.

Usage

```
epg_detect_xsd_for_gml(gml_path)
```

Arguments

<code>gml_path</code>	Path to a GML file.
-----------------------	---------------------

Value

Path to the XSD file, or `NA_character_` if none found.

epg_dfd_auth_params	<i>Datafordeler authentication query parameters</i>
---------------------	---

Description

Returns a named list of credential query parameters for Datafordeler URLs. Prefers API-key auth (apikey=) when DFD_API_KEY is set; falls back to legacy username/password (username=. . . &password=. . .). Errors if neither is available.

Usage

```
epg_dfd_auth_params(  
  api_key = Sys.getenv("DFD_API_KEY", ""),  
  username = Sys.getenv("DFD_USERNAME", ""),  
  password = Sys.getenv("DFD_PASSWORD", "")  
)
```

Arguments

api_key	API key (default Sys.getenv("DFD_API_KEY"))
username	Legacy service-user name (default Sys.getenv("DFD_USERNAME"))
password	Legacy service-user password (default Sys.getenv("DFD_PASSWORD"))

Value

Named list with either apikey OR username + password.

epg_dfd_auth_qs	<i>Datafordeler authentication query string</i>
-----------------	---

Description

Returns a URL-encoded query-string fragment for the credentials returned by [epg_dfd_auth_params\(\)](#), e.g. "apikey=abc123" or "username=u&password=p". Does not include a leading ? or &.

Usage

```
epg_dfd_auth_qs(...)
```

Arguments

...	Passed to epg_dfd_auth_params() .
-----	---

Value

Character scalar.

epg_drop_version	<i>Remove a dataset version and its data</i>
------------------	--

Description

Drops the version's partition (`vector.<dataset>_v<id>`), any staging table a failed run left behind, and its `meta.dataset_version` row, all in one transaction. Use it to take a bad load — truncated, duplicated, loaded with a stale loader — out of the warehouse before re-loading the same tag.

Usage

```
epg_drop_version(
  con,
  topic,
  dataset,
  version_tag,
  on_latest = c("error", "previous", "clear")
)
```

Arguments

<code>con</code>	A DBI connection.
<code>topic</code>	Topic name.
<code>dataset</code>	Dataset name.
<code>version_tag</code>	Tag of the version to remove.
<code>on_latest</code>	What to do when the version is the latest one: • "error" (default) refuses, leaving everything untouched. • "previous" moves the latest flag to the most recently loaded remaining version, via <code>meta.set_latest()</code> so its policies still apply. Errors if no other version exists. • "clear" leaves the dataset with no latest version, so <code>_latest</code> is empty until the next load.

Details

If rows for the version survive anywhere in `vector.<dataset>` (a partition not named by the pipeline's convention, say), the foreign key from the data to `meta.dataset_version` refuses the delete and the whole transaction rolls back, so a version is never removed from the catalog while its rows remain.

Removing the version that `vector.<dataset>_latest` serves would leave that view empty, so `on_latest` must say what happens instead.

COGs in `raster.cog_asset` are kept: that table records its source version without a foreign key, so a COG stays a valid historical artefact. They are reported, so you know to rebuild them.

Value

A list with `version_id`, `rows` (rows removed), `was_latest`, and `latest` (the tag the latest flag was moved to, or NA if it was not moved), invisibly.

<code>epg_ensure_dataset</code>	<i>Ensure a dataset row exists in meta.dataset</i>
---------------------------------	--

Description

Errors if topic does not exist, rather than silently inserting nothing and letting a later step fail with an opaque message.

Usage

```
epg_ensure_dataset(con, topic, dataset)
```

Arguments

<code>con</code>	A DBI connection.
<code>topic</code>	Topic name (must already exist).
<code>dataset</code>	Dataset name.

<code>epg_ensure_topic</code>	<i>Ensure a topic row exists in meta.topic</i>
-------------------------------	--

Description

Ensure a topic row exists in meta.topic

Usage

```
epg_ensure_topic(con, topic, description = NULL)
```

Arguments

<code>con</code>	A DBI connection.
<code>topic</code>	Topic name.
<code>description</code>	Optional topic description.

`epg_finalize_version` *Finalize a dataset version: mark latest and drop staging*

Description

Finalize a dataset version: mark latest and drop staging

Usage

```
epg_finalize_version(
  con,
  topic,
  dataset,
  version_tag,
  stage_tbl,
  set_latest = TRUE
)
```

Arguments

<code>con</code>	A DBI connection.
<code>topic</code>	Topic name.
<code>dataset</code>	Dataset name.
<code>version_tag</code>	The version tag string.
<code>stage_tbl</code>	The qualified staging table name to drop.
<code>set_latest</code>	Whether to mark this version as latest. Default TRUE.

`epg_fix_encoding` *Repair text that was decoded as UTF-8 but is actually Latin-1*

Description

Applies `iconv()` only to strings that are not already valid UTF-8. Converting unconditionally double-encodes correct UTF-8 — `ø` (C3 B8) becomes `Ãø` — and because Latin-1 has no invalid bytes, `iconv()` never errors to warn you. It just corrupts.

Usage

```
epg_fix_encoding(x, from = "latin1")
```

Arguments

<code>x</code>	A character vector.
<code>from</code>	Source encoding assumed for invalid strings.

Value

x with invalid entries re-decoded.

epg_insert_normalized *Insert normalized geometries from staging into a vector partition*

Description

Insert normalized geometries from staging into a vector partition

Usage

```
epg_insert_normalized(
  con,
  dataset,
  ver_id,
  stage_tbl,
  geom_transform =
    "ST_Multi(ST_CollectionExtract(ST_MakeValid(s.geom), 3))::geometry(MultiPolygon,25832)",
  delete_before_insert = FALSE
)
```

Arguments

con	A DBI connection.
dataset	Dataset name (parent table is vector.<dataset>).
ver_id	The dataset_version.id.
stage_tbl	The qualified staging table name (e.g., "stage.foo_s_42").
geom_transform	SQL expression for geometry normalization. Defaults to the standard Multi-Polygon transform. Use "ST_Force2D(ST_MakeValid(s.geom))::geometry(Geometry,25832)" for geodk.
delete_before_insert	If TRUE, deletes existing rows for this ver_id before inserting (for idempotent re-loads like LBST). Default FALSE.

Value

The number of rows inserted (invisible).

epg_parse_arg	<i>Parse a command-line flag argument</i>
---------------	---

Description

Returns the value following flag. A flag that is absent yields default. A flag that is present but malformed — repeated, or given without a value — is an error rather than a silent fallback to default, because these loaders run unattended and a typo must not quietly load with defaults.

Usage

```
epg_parse_arg(flag, default = NULL, args = commandArgs(trailingOnly = TRUE))
```

Arguments

flag	The flag string (e.g., "-src-url").
default	Default value if flag is not found.
args	Character vector of arguments. Defaults to <code>commandArgs(trailingOnly = TRUE)</code> .

Value

The value following the flag, or default when the flag is absent.

epg_parse_json_arg	<i>Parse a JSON value from a string or @file reference</i>
--------------------	--

Description

If value starts with @, the remainder is treated as a file path whose contents are read and parsed as JSON.

Usage

```
epg_parse_json_arg(value, label)
```

Arguments

value	A JSON string or @path/to/file.json.
label	A human-readable label used in error messages.

Value

Parsed JSON as an R list, or NULL if value is empty/NULL.

`epg_parse_json_payload`*Parse a JSON string*

Description

Parse a JSON string

Usage

```
epg_parse_json_payload(payload, label)
```

Arguments

<code>payload</code>	A JSON string.
<code>label</code>	A human-readable label used in error messages.

Value

Parsed JSON as an R list.

`epg_persist_source_meta`*Persist source metadata on a dataset version*

Description

Persist source metadata on a dataset version

Usage

```
epg_persist_source_meta(con, ver_id, source_meta)
```

Arguments

<code>con</code>	A DBI connection.
<code>ver_id</code>	The <code>dataset_version.id</code> .
<code>source_meta</code>	A named list of metadata to store as JSONB.

epg_pg_connect	Connect to PostgreSQL using PG* environment variables
----------------	---

Description

Reads PGHOST, PGPORT, PGDATABASE, PGUSER, PGPASSWORD from the environment. PGDATABASE, PGUSER and PGPASSWORD are required: this package is shared across warehouses, so guessing a database or falling back to a default superuser password risks connecting somewhere unintended.

Usage

```
epg_pg_connect(application_name = NULL)
```

Arguments

application_name
Optional application name tag for the connection (visible in pg_stat_activity).

Value

A PqConnection object.

epg_promote_latest	Promote the newest eligible year-vintage version to latest
--------------------	--

Description

Year-vintage datasets have several compose services sharing one dataset name and differing only by version tag (2024, 2025, 2026). Loading an older vintage must not leave it holding is_latest.

Usage

```
epg_promote_latest(con, topic, dataset, year = format(Sys.Date(), "%Y"))
```

Arguments

con A DBI connection.
topic Topic name.
dataset Dataset name.
year The eligible vintage, as a 4-digit string. A version qualifies when its tag is exactly year or begins year-.

Details

The rule is "the newest eligible vintage wins". Crucially, when no eligible version exists — which is the situation every 1 January, before the new year's data is published — the current flag is left alone. Clearing it instead would leave `vector.<dataset>_latest` returning zero rows, silently, until someone published a new vintage.

The flag itself is moved by `meta.set_latest()`, so the database keeps a single implementation of what "latest" means.

Value

The version tag now flagged latest, or `NA_character_` if the flag was left as it was.

<code>epg_read_json_file</code>	<i>Read and parse a JSON file</i>
---------------------------------	-----------------------------------

Description

Read and parse a JSON file

Usage

```
epg_read_json_file(path, label)
```

Arguments

<code>path</code>	Path to a JSON file.
<code>label</code>	A human-readable label used in error messages.

Value

Parsed JSON as an R list, or NULL if path is empty/NULL.

<code>epg_redact_url</code>	<i>Redact credentials from a URL</i>
-----------------------------	--------------------------------------

Description

Replaces the value of any known credential query parameter with REDACTED, and strips userinfo from the authority. Use this on every URL that reaches a log message, an error, or `meta.dataset_version`.

Usage

```
epg_redact_url(url)
```

Arguments

`url` A character vector of URLs.

Details

Unparseable input is reduced to its path, never returned verbatim, so a malformed URL cannot leak a secret through this function.

Value

`url` with credential values replaced by REDACTED.

`epg_register_version` *Register a new dataset version*

Description

Register a new dataset version

Usage

```
epg_register_version(
  con,
  topic,
  dataset,
  version_tag,
  on_conflict = c("error", "reuse")
)
```

Arguments

<code>con</code>	A DBI connection.
<code>topic</code>	Topic name.
<code>dataset</code>	Dataset name.
<code>version_tag</code>	Version tag string.
<code>on_conflict</code>	"error" (default) to fail on duplicate, or "reuse" to return the existing version id.

Reusing a version id means the caller is re-loading into a partition that may already hold rows. Callers that pass "reuse" must clear those rows first — see the `delete_before_insert` argument of [epg_insert_normalized\(\)](#). [epg_run_pipeline\(\)](#) enforces this pairing.

Value

The `dataset_version.id` (integer).

`epg_resolve_s3_config` *Resolve S3-compatible object storage configuration from environment*

Description

Reads `RAW_*` env vars, falling back to `SCW_*` for backward compatibility.

Usage

```
epg_resolve_s3_config(default_prefix = "")
```

Arguments

`default_prefix` Default S3 key prefix if `RAW_PREFIX` / `SCW_RAW_PREFIX` is not set (e.g., "raw/dataset/version").

Value

A named list with components `endpoint`, `region`, `bucket`, `prefix`, `access_key`, `secret_key`, and `configured` (logical).

`epg_run_pipeline` *Run the full ingest pipeline in a single transaction*

Description

Convenience wrapper that calls `epg_ensure_topic()`, `epg_ensure_dataset()`, `epg_register_version()`, `epg_set_raw_file_url()`, `epg_persist_source_meta()`, `epg_create_partition()`, `epg_write_staging()`, `epg_insert_normalized()`, and `epg_finalize_version()` inside `dbWithTransaction()`.

Usage

```
epg_run_pipeline(
  con,
  sf_obj,
  topic,
  dataset,
  version_tag,
  topic_description = NULL,
  source_meta = NULL,
  raw_url = NA_character_,
  geom_type = "MultiPolygon",
  geom_transform =
    "ST_Multi(ST_CollectionExtract(ST_MakeValid(s.geom), 3))::geometry(MultiPolygon,25832)",
  delete_before_insert = FALSE,
  set_latest = TRUE,
  on_conflict_version = "error",
```

```

    expected_features = NA_integer_,
    count_tolerance = 0,
    allow_empty = FALSE
  )

```

Arguments

<code>con</code>	A DBI connection.
<code>sf_obj</code>	An sf object with the features to load.
<code>topic</code>	Topic name.
<code>dataset</code>	Dataset name.
<code>version_tag</code>	Version tag string.
<code>topic_description</code>	Optional topic description.
<code>source_meta</code>	Named list of source metadata (or NULL). Any URL it contains is redacted before it is stored.
<code>raw_url</code>	Optional S3 URI of the raw file.
<code>geom_type</code>	Geometry type for the partition. Default "MultiPolygon".
<code>geom_transform</code>	SQL expression for geometry normalization.
<code>delete_before_insert</code>	If TRUE, deletes existing rows for this version before inserting. Required when <code>on_conflict_version = "reuse"</code> .
<code>set_latest</code>	Whether to mark this version as latest.
<code>on_conflict_version</code>	Passed as <code>on_conflict</code> to epg_register_version() .
<code>expected_features</code>	Feature count reported by the source, checked against the number of rows inserted before the version is published. NA (default) warns instead of checking.
<code>count_tolerance</code>	Fraction of <code>expected_features</code> that may be missing.
<code>allow_empty</code>	If FALSE (default), refuse to publish a version with no features — an empty upstream response must not blank out <code>_latest</code> .

Details

Everything that touches the database happens inside the transaction, so a failure at any step leaves no registered version, no partition and no half-loaded data. Do the download *before* calling this.

Value

The `dataset_version.id` (invisible).

epg_sanitize_name	<i>Sanitize a string into a valid SQL/table name</i>
-------------------	--

Description

Lowercases, replaces non-alphanumeric characters with underscores, and strips leading/trailing underscores.

Usage

```
epg_sanitize_name(x)
```

Arguments

x	A character string.
---	---------------------

Value

A sanitized character string.

epg_set_raw_file_url	<i>Set the raw file URL on a dataset version</i>
----------------------	--

Description

Credentials are stripped before the URL is stored: meta is readable by warehouse_reader, so a query string must never reach this column.

Usage

```
epg_set_raw_file_url(con, ver_id, raw_url)
```

Arguments

con	A DBI connection.
ver_id	The dataset_version.id.
raw_url	The S3 URI of the raw file. A length > 1 vector is an error; pass the single URI the version should record.

epg_shapefile_encoding_opts

GDAL open options for reading a shapefile's attributes correctly

Description

Danish sources ship shapefiles both as Latin-1 and as UTF-8. Forcing ENCODING=ISO-8859-1 overrides a .cpg sidecar that says otherwise, and silently mojibakes every æ, ø and å into the attr\$jsonb — where nothing downstream can tell it from real data.

Usage

```
epg_shapefile_encoding_opts(shp_path, fallback = "ISO-8859-1")
```

Arguments

shp_path	Path to a .shp file.
fallback	Encoding assumed when no .cpg is present.

Details

So: trust an explicit .cpg declaration when there is one, and only assume Latin-1 when the shapefile declares nothing.

Value

A character vector of GDAL open options for `sf::st_read()`.

epg_upload_raw_files *Upload raw files to S3-compatible object storage*

Description

Upload raw files to S3-compatible object storage

Usage

```
epg_upload_raw_files(files, s3_config, on_error = c("stop", "warn"))
```

Arguments

files	Character vector of local file paths to upload.
s3_config	A list as returned by <code>epg_resolve_s3_config()</code> .
on_error	"stop" (default) to abort on upload failure, or "warn" to emit a warning and continue.

Value

The S3 URI(s) of uploaded files (character vector), or NA_character_ for files that failed when on_error = "warn".

epg_url_without_query *Strip the query string from a URL*

Description

Returns the scheme, authority and path only. Use for the source_url recorded in meta.dataset_version.source_meta when the query string carries credentials.

Usage

```
epg_url_without_query(url)
```

Arguments

url A character vector of URLs.

Value

url without its query string or fragment.

epg_write_staging *Write an sf object to a staging table*

Description

Creates the stage schema if needed, writes the sf object via st_write, verifies creation, normalizes the geometry column name to geom, and returns a row count.

Usage

```
epg_write_staging(con, sf_obj, dataset, ver_id)
```

Arguments

con A DBI connection.
 sf_obj An sf object to stage.
 dataset Dataset name.
 ver_id The dataset_version.id (integer).

Value

A list with stage_tbl (qualified name like "stage.foo_s_42") and n_rows (count of rows with non-NULL geometry).

Index

epg_build_source_meta, 2
epg_check_feature_count, 3
epg_collect_source_meta_overrides, 3
epg_create_partition, 4
epg_create_partition(), 15
epg_detect_xsd_for_gml, 4
epg_dfd_auth_params, 5
epg_dfd_auth_params(), 5
epg_dfd_auth_qs, 5
epg_drop_version, 6
epg_ensure_dataset, 7
epg_ensure_dataset(), 15
epg_ensure_topic, 7
epg_ensure_topic(), 15
epg_finalize_version, 8
epg_finalize_version(), 15
epg_fix_encoding, 8
epg_insert_normalized, 9
epg_insert_normalized(), 14, 15
epg_parse_arg, 10
epg_parse_json_arg, 10
epg_parse_json_payload, 11
epg_persist_source_meta, 11
epg_persist_source_meta(), 15
epg_pg_connect, 12
epg_promote_latest, 12
epg_read_json_file, 13
epg_redact_url, 13
epg_register_version, 14
epg_register_version(), 15, 16
epg_resolve_s3_config, 15
epg_resolve_s3_config(), 18
epg_run_pipeline, 15
epg_run_pipeline(), 14
epg_sanitize_name, 17
epg_set_raw_file_url, 17
epg_set_raw_file_url(), 15
epg_shapefile_encoding_opts, 18
epg_upload_raw_files, 18
epg_url_without_query, 19
epg_write_staging, 19
epg_write_staging(), 15
sf::st_read(), 18