GreenieRE & Reinsurance Analytics

Phase 1 Technical Architecture

MGA Bordereaux Automation, LIDAC Eligibility, and ZIP Accumulation Risk

---

Objectives

Phase 1 delivers an end–to–end data pipeline that turns raw carrier and MGA bordereaux into:

- **Clean, standardized project records** across multiple carriers.
- **ZIP and census–tract level intelligence** using HUD crosswalks.
- **LIDAC / CEJST-style eligibility tags** at the tract level.
- **Intacct-ready journal entries** for premiums, commissions, and fees.
- **ZIP-level accumulation flags** to highlight clustering of exposure across carriers.
- **Clear export files** that Wenhua can inspect, audit, and load into existing systems.

The design is intentionally simple: one repeatable pipeline that can run for any MGA or carrier that supplies a bordereaux file with the current column structure.

System Architecture Diagram (Phase 1)

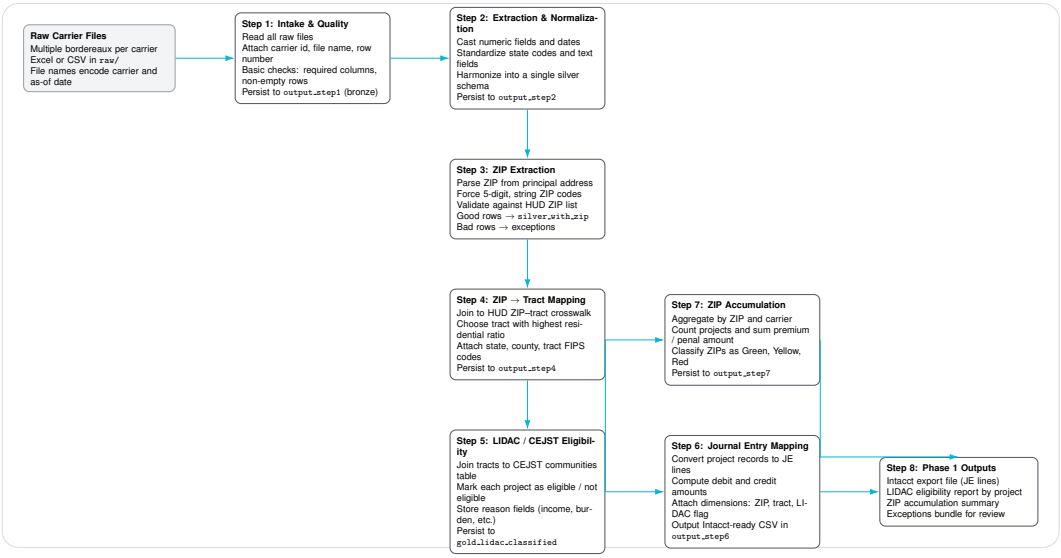

Figure 1: Phase 1 end-to-end pipeline from raw bordereaux files to accounting and risk outputs.

Pipeline Steps (Business-Facing Description)

Step 1 – Intake and Data Quality (Bronze Layer)

What runs. The `step1_intake_and_quality.py` script scans the `raw/` folder, reads every carrier file (CSV or Excel), and appends:

- Carrier identifier (parsed from file name, e.g. C001, C002, C003).
- Source file name and row number for full lineage.
- Ingestion timestamp in UTC.

Why it matters. This establishes a single, auditable "bronze" table where every bordereaux row can be traced back to the original file and line. If a number looks wrong later, we can immediately see which carrier file it came from.

Step 2 – Extraction and Normalization (Silver Layer)

What runs. The `step2_extraction_normalization.py` script:

- Casts numeric columns (*Gross Premium*, *Net Premium*, *Commission*, *Penal Amount*) into clean decimals.
- Converts dates (*Effective*, *Expiration*, *As-of Date*) into ISO date format.
- Normalizes state codes (e.g. "Pa" and "PA" both become "PA").
- Trims whitespace, cleans text, and applies a consistent column schema defined in `schema_registry.py`.

Why it matters. The result is a single, standardized "silver" table across all carriers, ready for geo-joining, eligibility tagging, and accounting logic.

Step 3 – ZIP Extraction and Validation

What runs. The `step3_zip_extraction.py` script:

- Parses the ZIP code out of the *Principal / Account Mailing Address*.
- Ensures the ZIP is always stored as a 5-character string (e.g. "02115", not "2115").
- Validates the ZIP against the HUD ZIP list and marks: *zip_valid_flag* and *zip_error_reason*.
- Routes invalid or missing ZIPs into an exceptions file.

Why it matters. Clean and validated ZIP codes are the foundation for all tract-level and LIDAC calculations. Fixing them here prevents downstream errors.

Step 4 – ZIP to Census Tract Mapping

What runs. The `step4_zip_to_tract_mapping.py` script:

- Joins each ZIP to the HUD ZIP–tract crosswalk.
- Selects the tract with the highest residential ratio when a ZIP maps to multiple tracts.
- Adds *state_fips*, *county_fips*, and *tract_fips* codes, plus a match ratio and flag.
- Routes any rows that cannot be matched into an exceptions file.

Why it matters. This step moves the data from postal geography (ZIPs) to statistical geography (census tracts), which is the level used by CEJST and other federal programs.

Step 5 – LIDAC / CEJST Eligibility Classification

What runs. The `step5_lidac_eligibility_cejst.py` script:

- Loads the official CEJST communities table (`cejst_v2_communities.csv`).
- Joins project tracts to CEJST tracts.
- Tags each project with: *cejst_disadvantaged*, *lidac_eligible*, and *lidac_reason*.
- Writes both a main classified file and a list of projects that could not be matched (for manual review).

Why it matters. GreenieRE can now answer, for every project, whether it sits in a disadvantaged/eligible tract and why, in a way that is transparent and defensible to regulators and capital providers.

Step 6 – Journal Entry Mapping for Intacct

What runs. The `step6_journal_entry_mapping.py` script:

- Transforms each project into 2–3 journal entry lines using a configurable accounting template.
- Calculates debits and credits for premium, commission, and vendor fees.
- Attaches project dimensions (carrier, product, state, ZIP, tract, LIDAC flag) so Intacct can filter by geography and eligibility.
- Writes a clean `gold_journal_entries_for_intacct.csv` file.

Why it matters. This closes the loop from underwriting to accounting, providing a reconciliation-ready feed that can be reviewed and then loaded into Intacct.

Step 7 – ZIP-Level Accumulation Detection

What runs. The `step7_zip_accumulation.py` script:

- Aggregates project records by ZIP code and carrier.
- Computes project counts, total gross premium, total penal amount, and the number of distinct carriers in each ZIP.
- Applies thresholds to flag ZIP codes as Green, Yellow, or Red based on density and premium concentration.
- Produces a concise ZIP accumulation table for reinsurer risk review.

Why it matters. This addresses James's concern about hidden stacking of risk across MGAs in the same ZIP and surfaces it in a simple, quantitative way.

Step 8 – Phase 1 Output Packaging

What runs. The `step8_phase1_outputs.py` script:

- Bundles Intacct journal entries, LIDAC eligibility outputs, ZIP accumulation tables, and all exceptions into a single place.
- Adds basic logging (row counts, file paths) so Wenhua can confirm what was produced each time the pipeline runs.

Why it matters. GreenieRE receives a small, predictable set of deliverables every run: one accounting file, one eligibility file, one accumulation file, and one exceptions summary. This makes Phase 1 easy to explain, test, and scale.

Summary. Phase 1 implements a complete, auditable pipeline from raw carrier bordereaux to tract-level LIDAC eligibility, Intacct-ready journal entries, and ZIP-level accumulation alerts, using only freely available public data and the current column structure Wenhua has already shared.