**COBOL OVERVIEW**

This document provides a high-level, plain English description of the COBOL program **EDBNAECE** based on the supplied source code (from "EDBNAECE.SRB"), business requirements (from "Business Requirements11.txt"), and input/output file layouts (from "Cobol Output-1.xlsx"). The focus is on **what the program does** and **how it does it**, including its overall purpose, inputs, outputs, processing logic, and key business rules. This serves as a baseline for comparison across tools or POCs, without delving into code conversion details (e.g., to .NET).

The description is derived from analyzing the program's structure, sections, file handling, data accumulation, conditional logic, and external module calls. It avoids technical jargon where possible, explaining concepts in simple terms.

## 1. Program Overview

### What the Program Does

* **Core Purpose**: This program is part of the "NAS Evergreening" system, which appears to be a data quality and maintenance process (likely for address or entity data in a financial or compliance context). It reads input files containing audit logs and "DIN" records (DIN seems to stand for something like "Data Identification Number" – unique identifiers for data entities). The program counts and aggregates key metrics from these inputs, such as the number of DINs, PINs (possibly "Primary Identification Numbers"), LINs (possibly "Linked Identification Numbers"), and errored records. It then updates a statistics file with these totals and generates a detailed "display statistics" file for reporting. In "update mode," it also creates a "reapply" file to flag DIN records that need further processing or correction.
* **Business Role**: It acts as a batch summarizer and reporter. By tracking metrics like data quality changes (e.g., from low-quality to high-quality addresses) and errors, it helps monitor data processing health. The reapply file supports downstream workflows, like reprocessing faulty data. This fits into broader business goals of ensuring data integrity, tracking volumes, and enabling corrections without manual intervention.
* **When It's Run**: Called by a job (NENAS050) in a mainframe batch environment. It runs in one of two modes: "Read" (just analyzes and reports) or "Update" (analyzes, reports, and creates the reapply file).
* **Key Outcomes**:
  + Accurate counts of processed data items (DINs, PINs, LINs).
  + Detailed reports on data quality metrics (e.g., unchanged addresses, quality upgrades).
  + Logging for auditing and troubleshooting.
  + Reapply file for fixes (only in update mode).
  + No database interactions (no DB2 tables used; all file-based).

### How It Fits into the Business

From the business requirements, this program supports data monitoring in NAS Evergreening by aggregating metrics and enabling reapplication. It assumes valid input data and relies on external modules for logging and time tracking. Constraints include strict mode validation (only 'R' or 'U' allowed) and no internet/package installation access.

## 2. Inputs Expected

The program expects specific input files and parameters. If inputs are invalid (e.g., wrong mode or non-numeric logging level), it abends (crashes with an error message).

### Input Files

#### Audit Input File (DDNAME: AUDITI)

* **Description**: Contains processed audit records from NAS Evergreening. Each record includes flags for errors/processing status, counts of PINs and non-standard LINs, timestamps, DINs, source data (e.g., names, addresses, quality codes like 'H' for High, 'L' for Low, 'T' for Text), error codes, and arrays for multiple PINs/LINs/DINs.
* **Format/Layout** (from Excel "Input File Layout" sheet):
  + Variable-length records (recording mode V).
  + Key fields: Process stage (error/processed flags), PIN count (S9(4) COMP), Non-std LIN count (S9(4) COMP), DIN (S9(18) COMP), Source Address Quality (X(01)), Error Code (X(04)), arrays for PINs (up to 500), LINs (up to 200), and DIN reapplies (up to 2200).
  + Example Data (from Excel): Records like ID=1 with IsError=0, IsProcessed=1, PinCount=0, NonStdLinCount=0.
* **How Used**: Read sequentially to accumulate PIN/LIN/error counts.

#### DIN Input File (DDNAME: DININP)

* **Description**: Contains DIN records with subject numbers and sequences. Used to count DINs and (in update mode) generate reapply records.
* **Format/Layout** (from Excel "Input File Layout" sheet):
  + Fixed-length records (recording mode F).
  + Key fields: DIN (S9(18) COMP), Subject Number (S9(04) COMP), Subject Sequence Number (S9(04) COMP).
  + Example Data (from Excel): DIN=2010070010551797186, Subject Nb=1, Subject Seq Nb=2.
* **How Used**: Read sequentially to increment DIN count and write to reapply file if needed.

#### Statistics File (DDNAME: STATSO)

* **Description**: An input-output file read at start for existing stats and updated at end with new totals.
* **Format/Layout**: Fixed-length (X(287) or similar), with fields like Reapply DIN Count (S9(9) COMP), PIN Count, LIN Deleted, Total Errors, CPU Time, Elapsed Time.
* **How Used**: Read once at initialization; rewritten at termination.

### Parameters (Passed via Linkage Section)

* **Call Mode**: 'U' (Update – creates reapply file) or 'R' (Read – no reapply file). Invalid modes cause abend.
* **Logging Level**: Numeric value (e.g., 0001+ for detailed logs). Non-numeric causes abend.
* **Log Length**: Used for logging module.

## 3. Outputs Created or Updated

### Output Files

#### Reapply Output File (DDNAME: REAPPLY)

* **Description**: Only created in update mode. Contains DIN records flagged for reapplication (e.g., for data fixes). Each record includes source process, run date, DIN, number of subjects, and arrays for subject details.
* **Format/Layout** (from Excel "Output File Layout" sheet):
  + Variable-length records (recording mode V).
  + Key fields: Source Process (9(06)), Run Date (X(08)), DIN (S9(18) COMP), No of Subjects (9(01)), Arrays for Subject Nb/Seq (up to 9 occurrences).
  + Example Data (from Excel): Records like ID=1, DIN=2010070010551797186, SrcProcess=10, RunDate=45916, NoOfSubj=1, SubjNb=[1], SubjSeq=[2].
* **How Created**: Written sequentially for each unique DIN from the DIN input file (avoids duplicates by tracking previous DIN).

#### Detailed Statistics File (DDNAME: STATSD)

* **Description**: A human-readable report with headers and counts for monitoring (e.g., low-quality records, errored records, quality changes).
* **Format/Layout** (from Excel "Output File Layout" sheet):
  + Fixed-length records (X(48) or similar).
  + Key fields: Header (X(40), e.g., "NUMBER OF LOW QUALITY INPUT RECORDS :"), Details (Z(10)9(7) – zero-suppressed numeric).
  + Example Data (from Excel): Lines like "NUMBER OF ERRORED RECORDS : 1", "TOTAL NUMBER OF INPUT RECORDS : 3".
* **How Created**: Written at termination with predefined headers and accumulated counts.
* **Statistics File (DDNAME: STATSO)**: Updated with final totals (e.g., reapply DIN count, PIN count, errors, CPU/elapsed time).

### Other Outputs

* **Logs**: Via external module ET530. Logs entry/exit if logging level > 0, plus errors/abends.
* **Displays/Abends**: Console messages for errors (e.g., "EDBNAECE IS ABENDING" with details).

## 4. Processing Logic: How the Program Works

The program follows a structured flow: setup (initialization), core processing (loop through inputs), and cleanup (termination). It uses flags (e.g., end-of-file indicators) and accumulators (counters for DIN/PIN/LIN/errors) to track state.

### Step-by-Step Flow

#### 1. Initialization (A0000-INITIALIZATION Section)

* Accepts parameters (call mode, logging level) and validates them (abends if invalid).
* Calls logging module (ET530) to log program start if logging level > 0.
* Records start time (via E1902 module for CPU/elapsed time).
* Opens files: Audit and DIN as input, Stats as input-output, Detailed Stats and Reapply as output.
* Reads the initial stats file record to load existing totals.
* Reads the first audit and DIN records.

#### 2. Main Processing (B0000-MAIN-PROCESS Section)

* Loops until end of audit file:
  + Accumulates from audit records: Adds PIN count, LIN count, and increments error count if record is errored.
  + Reads next audit record.
* Loops until end of DIN file:
  + Increments DIN count.
  + In update mode: Checks if DIN is new (compares to previous); if so, formats and writes a reapply record (includes DIN, subjects, run date, etc.).
  + Reads next DIN record.
* Calls time calculation module (E1902) to get total CPU/elapsed time.

#### 3. Termination (C0000-TERMINATION Section)

* Updates stats file with final accumulators (e.g., total DINs, PINs, LINs, errors, times).
* Generates detailed stats file: Writes lines with headers and values (e.g., low-quality records, quality changes from business metrics).
* Closes files.
* Calls logging module to log program end if logging level > 0.
* Stops the program normally (or abends if errors occurred earlier).

### Key Business Rules in Logic

* **Modes**: Update mode writes reapply; Read mode skips it.
* **Duplicates**: Skips writing reapply for duplicate DINs (tracks previous DIN).
* **Error Handling**: Accumulates errored records; abends on file open/read errors or invalid params.
* **Metrics**: Tracks quality changes (e.g., low-to-high quality addresses, unchanged AINs) based on audit flags.
* **Logging**: Only if level > 0; includes program name, return codes, and messages.

### External Dependencies

* **ET530**: Logging module – called at start/end and for errors.
* **E1902**: Time calculation – gets CPU/elapsed times.
* No databases; all file-based.

## 5. Potential Improvements or Notes

* The program is efficient for batch processing but relies on sequential file reads (could be slow for very large files).
* Error recovery is basic (abends halt everything); modern versions could add retries.
* From business recommendations: Consider databases for scalability, better logging formats.