Disclaimer: This technical design specification document has been generated by a Large Language Model (LLM) and should be reviewed and validated by subject matter experts before use.

COBOL - Technical Design Specification for Modernization: EXWWB909

Program type: Batch

Total Lines of code: 3131

Total Tables: 12

Total files: 3

Last Modified: This code was last modified on 02/08/2018

Update Tracking

Sr. No.	Update Date	Remarks
1	2025-06-12	New document generated

Contents

1. Introduction

1.1 Purpose

The purpose of program EXWWB909 is to extract vehicle, customer, and dealer data from various GEVIS-owned DB2 tables. This extracted data is used to create an outbound bridge file for the VINCENT (North American Incentive Claiming System). VINCENT utilizes this file to re-review VRULES programs and compare them against VIN records, specifically for chargeback (automatic post-claim validation) and auto claim generation during a "full pass" in VRULES.

1.2 Scope

The scope of EXWWB909 includes:

- Reading an input SYSPARM file containing a list of producers.
- Processing one producer at a time.
- For each producer, selecting vehicle data from `MEXW001_VEH_ORDER` for model years -4 to +2 relative to the current year.
- Further processing and outputting data only for vehicles associated with a current WDMO dealer.
- Creating an outbound bridge file (`VINCENT-OUT`) with standard E&G headers/trailers and VINCENT headers/trailers.
- Handling cases where extracted fields might be unpopulated (spaces for alphanumeric, zeros for numeric).
- Managing run control using a last run timestamp stored in `MEXS016_GENERIC2`.

1.3 Audience

This document is intended for COBOL developers, system analysts, and testers involved in the maintenance, modernization, or understanding of the EXWWB909 program and its interactions within the GEVIS and VINCENT systems.

2. Overview

2.1 Background

EXWWB909 is a batch program critical for data synchronization and validation between the GEVIS and VINCENT systems. It ensures that VINCENT has up-to-date vehicle, customer, and dealer information for its incentive claiming and validation processes. The program is a clone of EXWWB910 and EXWWGEVP, implying a need for coordinated changes across these programs. A key modification involved changing model year criteria in cursors from -1/+2 to -4/+2 years.

2.2 Objectives

- To accurately extract specified data elements from GEVIS DB2 tables.
- To filter records based on producer list, model year range, and WDMO dealer association.
- To generate a correctly formatted outbound bridge file for VINCENT.
- To maintain processing integrity through run control timestamps and batch numbers.
- To provide audit trails for processing activities and errors.

2.3 Assumptions and Constraints

- The input SYSPARM file (`INPUT-PARM`) is correctly formatted and contains valid producer codes.
- Required DB2 tables (MEXW001, MEXW003, MEXW008, MEXS016, etc.) are accessible and contain the necessary data.
- The subprogram `EXWWSSTK` is available and functions as expected for retrieving dealer information.
- Copybooks defining data structures are accurate and available.
- The program operates in a batch environment with necessary JCL and system resources.
- If certain fields from DB2 are not populated, they will be output as spaces (alphanumeric) or zeros (numeric).

3. System Architecture

3.1 Component Diagram

Standardized Component Categories (use these exact labels):

1. **External Systems:** COMP_[SYSTEM_NAME]
2. **Databases:** DB_[TABLE_NAME]
3. **Files:** FILE_[FILENAME]
4. **Subprograms:** SUBPROG_[PROGRAM_NAME]
5. **Main Program:** MAIN_[PROGRAM_ID]

Required Connections (use these exact labels):

- Database access: MAIN –>|reads/writes| DB_TABLE
- File processing: MAIN –>|processes| FILE_NAME
- Program calls: MAIN –>|calls| SUBPROG_NAME

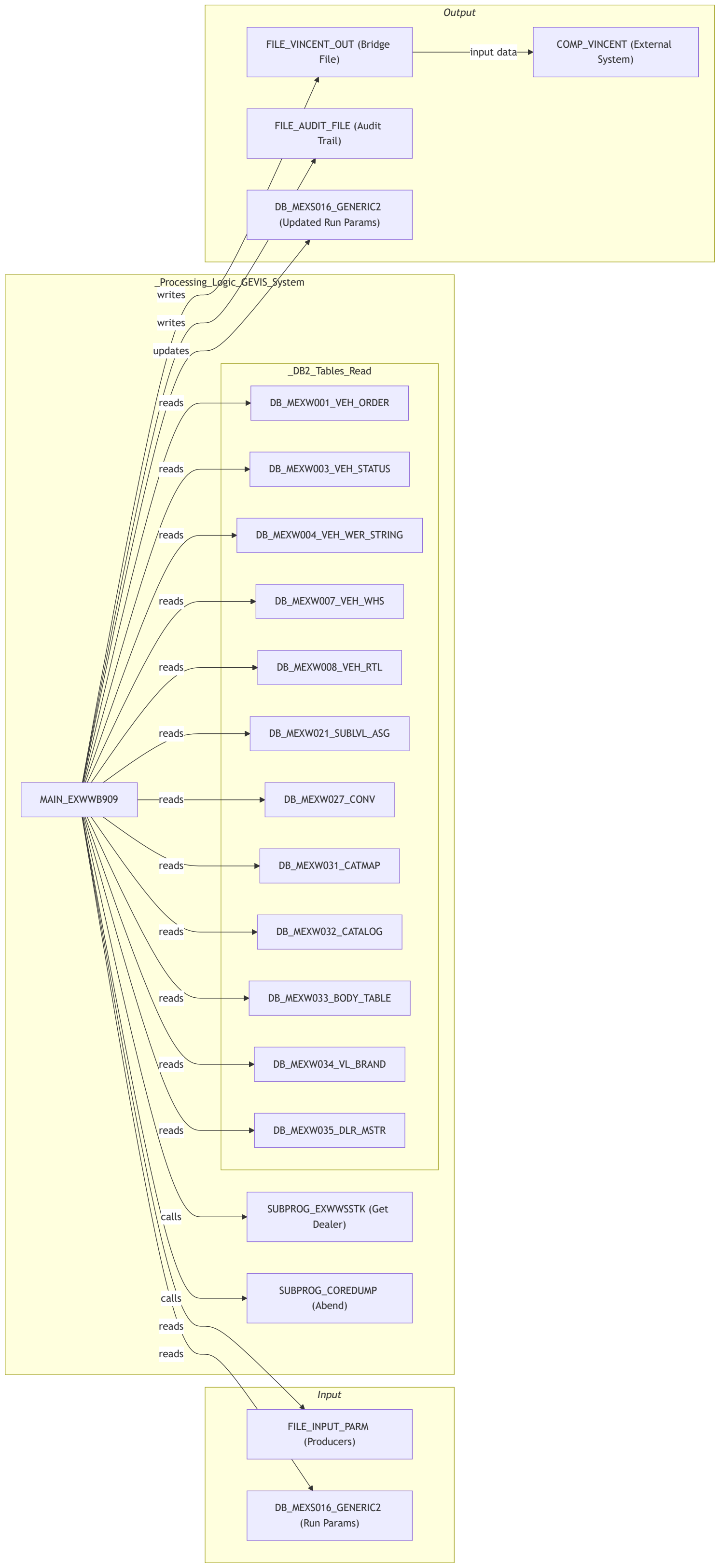

Figure 1: Diagram 1

3.2 Control Flow Diagram

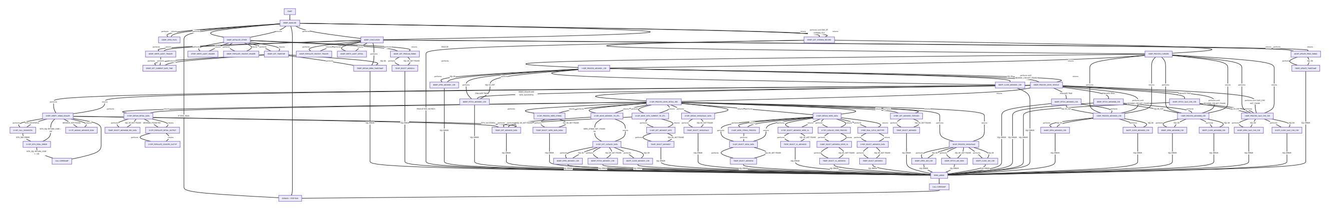

Figure 2: Diagram 2

4. Detailed Design

4.1 Program Structure

The program EXWWB909 operates in a sequential batch mode, structured into three main phases:

1. **Initialization Phase (`0000P-MAINLINE` performing `0300P-OPEN-FILES`, `0400P-INITIALIZE-OTHER`):**
   - Opens input (`INPUT-PARM`), output (`AUDIT-FILE`, `VINCENT-OUT`) files.
   - Initializes working storage areas, counters, and switches.
   - Retrieves the current system timestamp (`0500P-GET-TIMESTMP` via `7000P-OBTAIN-DRBN-TIMESTAMP`).
   - Fetches program control parameters (last run timestamp, previous batch number) from `MEXS016_GENERIC2` (`0600P-GET-PROGAM-PARMS` via `7010P-SELECT-MEXS016`). Calculates the current batch number.
   - Writes header records to the audit file (`0750P-WRITE-AUDIT-HEADER`) and the VINCENT output file (`0800P-POPULATE-VINCENT-HEADER`).
2. **Main Processing Phase (`0000P-MAINLINE` performing `0700P-GET-SYSPARM-RECORD` in a loop, which in turn performs `1000P-PROCESS-CURSORS`):**
   - Reads producer codes one by one from the `INPUT-PARM` file.
   - For each producer, it sequentially processes data through four main DB2 cursors:
     - `1100P-PROCESS-MEXW001-CSR`: Selects vehicles from `MEXW001_VEH_ORDER` based on update timestamp and model year.
     - `1200P-PROCESS-MEXW003-CSR`: Selects vehicles by joining `MEXW001_VEH_ORDER` and `MEXW003_VEH_STATUS` based on status update timestamp and model year.
     - `1300P-PROCESS-MEXW008-CSR`: Selects vehicles by joining `MEXW001_VEH_ORDER` and `MEXW008_VEH_RTL` based on retail update timestamp and model year.
     - `1400P-PROCESS-SALE-CHK-CSR`: Selects older vehicles (model year < current - 4) sold recently, joining `MEXW001_VEH_ORDER` and `MEXW008_VEH_RTL`.
   - Each vehicle fetched by these cursors undergoes further processing in `2000P-PROCESS-GEVIS-VEHICLE`:
     - `2105P-VERIFY-WDMO-DEALER`: Calls subprogram `EXWWSSTK` to get the stocking dealer and then checks `MEXW035_DLR_MSTR` to confirm if it's a WDMO dealer.
     - If the dealer is WDMO and `EXWWSSTK` call is successful, `2120P-PROCESS-GEVIS-DETAIL-REC` is performed:

        * Gathers detailed vehicle information by accessing various DB2 tables (`MEXW004`, `MEXW031`, `MEXW032`, `MEXW033`, `MEXW034`, `MEXW007`, `MEXW027`) and populates `WS-VINCENT-DETAIL-RECORD`.
        * This involves fetching WERS string, catalog data, retail sales data, wholesale data, and various status dates.
        * Writes the populated detail record to `VINCENT-OUT`.

3. **Termination Phase (`0000P-MAINLINE` performing `6000P-CONCLUSION`):**
    - Retrieves the current system timestamp again.
    - Updates program control parameters (current timestamp as the new last run timestamp, current batch number) in `MEXS016_GENERIC2` (`6010P-UPDATE-PROG-PARMS` via `7005P-UPDATE-TIMESTAMP`).
    - Writes trailer records to the VINCENT output file (`6030P-POPULATE-VINCENT-TRAILER`).
    - Writes summary statistics and trailer records to the audit file (`6040P-WRITE-AUDIT-DETAIL`, `6050P-WRITE-AUDIT-TRAILER`).
    - Closes all files.
    - Sets `RETURN-CODE` to 3 if `WS-SEND-EMAIL` flag is set (e.g., due to `EXWWSSTK` not finding a dealer).
    - Ends program execution (`GOBACK`).
    - Error handling is integrated throughout, with severe errors leading to `9999I-ABEND` which calls `COREDUMP`.

4.2 Data Structures

This section details data structures used for file I/O and subprogram calls.

- **`AUDIT-RECORD`**: Record layout for the `AUDIT-FILE`.
    - Purpose: Used to write audit trail messages, headers, and footers.
    - Layout: `COBOL    01  AUDIT-RECORD.       05  AUDIT-LABEL              PIC X(30).       05  AUDIT-DATA              PIC X(50).`
    - Copybooks Referenced: None (defined in `FILE SECTION`).
- **`PARM-RECORD`** / **`PARM-DETAIL`**: Record layouts for the `INPUT-PARM` file.
    - Purpose: `PARM-RECORD` defines the raw input from `INPUT-PARM`. `PARM-DETAIL` is a working-storage structure to receive the input producer data.
    - Layout (`PARM-RECORD` from `FILE SECTION`): `COBOL    01  PARM-RECORD.       05  PARM-DATA              PIC X(80).`

- Layout (`PARM-DETAIL` from `WORKING-STORAGE SECTION`): `COBOL    01 PARM-DETAIL.       05 INPUT-SOURCE                    PIC X(02).      05 FILLER PIC X(78).`
- Copybooks Referenced: None (defined in `FILE SECTION` and `WORKING-STORAGE SECTION`).

- **`VINCENT-RECORD`**: Record layout for the `VINCENT-OUT` file.
  - Purpose: Defines the output record structure for the bridge file sent to VINCENT.
  - Layout: `COBOL    01   VINCENT-RECORD.        05   VINCENT-DETAIL           PIC X(1000).`
  - Copybooks Referenced: None (defined in `FILE SECTION`).
- **`VIN-HEADER-TRAILER`**: Record layout for VINCENT header and trailer records.
  - Purpose: Used to format header and trailer records for the `VINCENT-OUT` file.
  - Layout: Defined in copybook `CPEWVNCT`. `COBOL    01 VIN-HEADER-TRAILER.       COPY CPEWVNCT.   *VINCENT HEADER/TRAILER LAYOUT`
  - Copybooks Referenced:
    - `CPEWVNCT`
- **`WS-VINCENT-DETAIL-RECORD`**: Working storage layout for the detail records written to `VINCENT-OUT`.
  - Purpose: Accumulates all extracted and processed vehicle, customer, and dealer data before writing to the output file.
  - Layout (Hierarchical Outline): `01   WS-VINCENT-DETAIL-RECORD.   05   WS-DTL-VIN-FULL-C   PIC X(17).   05   WS-DTL-DTA-DATA-SRC-C   PIC X(02).   05   WS-DTL-BDT-MDL-YR-Y   PIC X(02). 05   WS-DTL-GEVIS-VEH-LINE-C   PIC X(02).   05   WS-DTL-LCL-BDYTYP-C   PIC X(05).   05   WS-DTL-CUR-STA-STATUS-C   PIC X(03). 05   WS-DTL-VEH-DIVISION-C   PIC X(01).   05   WS-DTL-WMI-WMI-C PIC X(03).   05   WS-DTL-LCL-PLT-C   PIC X(03).   05   WS-DTL-VWS-TOT-US-A   PIC S9(07)V99 COMP-3.   05   WS-DTL-VEH-GBL-DLR-C   PIC X(06).   05   WS-DTL-LAST-QAD-VST-GBL-LOC-C   PIC X(06).   05   WS-DTL-CURR-VST-GBL-LOC-C   PIC X(06).   05   WS-DTL-SHIP-TO-DLR-C   PIC X(06).   05   WS-DTL-CURR-STOCKING-DLR-C PIC X(06).   05   WS-DTL-CURR-DLR-C   PIC X(06).   05 WS-DTL-WDMO-FLEET-C   PIC X(05).   05   WS-DTL-VRS-LCL-FLEET-C   PIC X(06).   05   WS-DTL-VRS-CST-FIRST-N   PIC X(30). 05   WS-DTL-VRS-CST-MID-INIT-X   PIC X(01).   05   WS-DTL-VRS-CST-LAST-N   PIC X(30).   05   WS-DTL-VRS-CST-ADDR-1-X   PIC X(40).`

```
05  WS-DTL-VRS-CST-ADD-DIV2-N       PIC X(40).       05  WS-DTL-VRS-CST-ADD-
DIV1-C       PIC X(02).       05  WS-DTL-VRS-CST-POSTAL-C          PIC X(10).
05  WS-DTL-VRS-SALESPERSON-C        PIC X(11).       05  WS-DTL-VRS-TYP-LCL-
CUST-C       PIC X(01).       05  WS-DTL-VEH-WDMO-ORD-TYP          PIC X(01).
05  WS-DTL-VEH-ORD-RCPT-Y           PIC X(08).       05  WS-DTL-VEH-SCHD-VST-
TARGET-Y    PIC X(08).       05  WS-DTL-VEH-PRODUCE-VST-STAT-Y   PIC X(08).
05  WS-DTL-VEH-RELEASE-VST-STAT-Y   PIC X(08).       05  WS-DTL-VEH-ARRIVAL-
VST-STAT-Y   PIC X(08).       05  WS-DTL-VEH-INVOICE-VST-STAT-Y   PIC X(08).
05  WS-DTL-VEH-STOCK-VST-STAT-Y     PIC X(08).       05  WS-DTL-VEH-RETAIL-
VST-STAT-Y    PIC X(08).       05  WS-DTL-VEH-DELIVER-VST-STAT-Y   PIC X(08).
05  WS-DTL-VEH-SLSRCPT-VST-STAT-Y   PIC X(08).       05  WS-DTL-VEH-WARRANT-
VST-STAT-Y   PIC X(08).       05  WS-DTL-VEH-CATALOG-C            PIC X(15).
05  WS-DTL-WERS-VEH-LINE-C          PIC X(02).       05  WS-DTL-WERS-BODY-
STYLE-C        PIC X(03).       05  WS-DTL-WERS-BRAND-C             PIC X(01).
05  WS-DTL-VEH-PO-Y                 PIC X(08).       05  WS-DTL-FILLER-01
PIC X(627) VALUE SPACES.
```

    - Copybooks Referenced: None (defined in `WORKING-STORAGE SECTION`).

- **`SSTK-I-O-DATA`**: Data structure used for calling subprogram `EXWWSSTK`.
    - Purpose: Passes input parameters (mode, data source, order ID) to `EXWWSSTK` and receives output (dealer information, status codes, error details).
    - Layout: Defined in copybook `CPEWSSTK`. `COBOL    01  SSTK-I-O-DATA.       05`

```
SSTK-INPUT-DATA.          10  SSTK-MODE                    PIC X(01).
88  SSTK-INQUIRY-MODE        VALUE "I".               88  SSTK-UPDATE-
MODE         VALUE "U".           10  SSTK-DTA-DATA-SRC-C          PIC X(02).
10   SSTK-VEH-ORD-ID-C          PIC X(25).       05  SSTK-OUTPUT-DATA.
10  SSTK-GBL-STK-DLR-C          PIC X(06).           10  SSTK-LCL-STK-DLR-
C            PIC X(07).           10  SSTK-STK-DLR-STAT-C          PIC X(03).
10  SSTK-STK-DLR-STAT-Y         PIC X(10).           10  SSTK-STK-DLR-CNTRY-
ISO3-C     PIC X(03).          10  SSTK-CUR-STAT-C              PIC X(03).
10  SSTK-CUR-LCL-STAT-C         PIC X(06).           10  SSTK-CUR-STAT-Y
PIC X(10).          10  SSTK-DIV-DIV-C               PIC X(02).       05
SSTK-OUT-DATA-MSG.          10  SSTK-PGM-ID                   PIC
X(08).          10  SSTK-RETURN-CD                 PIC X(01).             88
SSTK-SUCCESSFUL            VALUE "0".               88  SSTK-INPUT-ERROR
VALUE "1".            88  SSTK-DB2-ERROR               VALUE "2".            10
SSTK-PARAGRAPH                PIC X(06).           10  SSTK-DB2-AREA.
15  SSTK-HOST-VAR1            PIC X(80).                ... (SSTK-HOST-VAR2
TO SSTK-HOST-VAR8)              15  SSTK-DB2-TABLES.                   20
SSTK-DB2-TABLE1       PIC X(18).                    ... (SSTK-DB2-TABLE2
TO SSTK-DB2-TABLE5)               15  SSTK-SQL-FUNCTION        PIC X(12).
```

```
15  SSTK-SQL-RETURN-CODE     PIC S9(04) COMP-3.               15  SSTK-SQL-
WARNING         PIC X(08).               15  SSTK-SQL-ERROR-MESSAGE   PIC
X(70).              15  SSTK-SQL-FULL-ERROR.                        20  SSTK-
SQL-MSG1        PIC X(72).                   ... (SSTK-SQL-MSG2 TO SSTK-SQL-
MSG4)              15  SSTK-SQLCA                 PIC X(200).       05  SSTK-
FILLER                       PIC X(1596).
```

- Copybooks Referenced:
  - CPEWSSTK

4.3 Algorithms

4.3.1 Overall Program Logic (Condensed Pseudocode)

```
START
  PERFORM Initialize_Program
    Open Files (AUDIT-FILE, VINCENT-OUT, INPUT-PARM)
    Initialize Working Storage (Variables, Switches, Counters)
    Get_Current_Timestamp (WS-CURR-DRBN-TIMESTAMP)
    Get_Program_Parameters (WS-PREV-BATCH-NBR, WS-PREV-RUN-TIMESTAMP from MEXS016)
    Calculate WS-CURRENT-BATCH-NBR
    Write_Audit_Header
    Populate_And_Write_Vincent_Header (to VINCENT-OUT)

  READ INPUT-PARM record INTO PARM-DETAIL
  PERFORM UNTIL END-OF-SYSPARM-FILE
    IF first SYSPARM read AND no records
      ABEND program ("MISSING SYSPARM RECORDS")
    END-IF

    Set Producer_Data_Source from PARM-DETAIL.INPUT-SOURCE
    Initialize Cursor_Switches (MEXW001_CSR-NOT-FOUND, etc.)

    PERFORM Process_MEXW001_Cursor
      OPEN MEXW001_CSR
      FETCH MEXW001_CSR
      PERFORM UNTIL MEXW001_CSR_NOT_FOUND
        PERFORM Process_GEVIS_Vehicle_Record
        FETCH MEXW001_CSR
      END-PERFORM
      CLOSE MEXW001_CSR
    END-PERFORM
  PERFORM Process_MEXW003_Cursor
    OPEN MEXW003_CSR
    FETCH MEXW003_CSR
    PERFORM UNTIL MEXW003_CSR_NOT_FOUND
      PERFORM Process_GEVIS_Vehicle_Record
      FETCH MEXW003_CSR
    END-PERFORM
    CLOSE MEXW003_CSR
  END-PERFORM

  PERFORM Process_MEXW008_Cursor
    OPEN MEXW008_CSR
    FETCH MEXW008_CSR
    PERFORM UNTIL MEXW008_CSR_NOT_FOUND
      PERFORM Process_GEVIS_Vehicle_Record
      FETCH MEXW008_CSR
    END-PERFORM
    CLOSE MEXW008_CSR
  END-PERFORM

  PERFORM Process_SALE_CHK_Cursor
    OPEN SALE_CHK_CSR
    FETCH SALE_CHK_CSR
    PERFORM UNTIL SALE_CHK_NOT_FOUND
      PERFORM Process_GEVIS_Vehicle_Record
      FETCH SALE_CHK_CSR
    END-PERFORM
    CLOSE SALE_CHK_CSR
  END-PERFORM

  READ INPUT-PARM record
END-PERFORM

PERFORM Conclude_Program
  Get_Current_Timestamp
  Update_Program_Parameters (WS-HOLD-CURR-TIMESTAMP, WS-CURRENT-BATCH-NBR to MEXS016)
  Populate_And_Write_Vincent_Trailer (to VINCENT-OUT)
  Write_Audit_Detail_Counts
  Write_Audit_Trailer
  Close Files
  END-PERFORM

  IF SEND-EMAIL flag is TRUE
    Set RETURN-CODE = 3
  END-IF
  GOBACK.

Process_GEVIS_Vehicle_Record:
  CALL "EXWWSSTK" to get Stocking_Dealer_Info
  IF EXWWSSTK successful
    Read MEXW035_DLR_MSTR to check if Stocking_Dealer is WDMO
    IF WDMO_Dealer
      Process WERS string data (from MEXW004_VEH_WERS_STRING)
      Move MEXW001 data to WS-DTL fields
      IF WERS string not found, get catalog data (from MEXW031_CATMAP)
      Obtain Retail Data (join MEXW003_VEH_STATUS and MEXW008_VEH_RTL for 90V status)
      Populate Retail Output Fields (customer name, address, sales type, dates)
      Move EXWWSSTK current stocking dealer info to WS-DTL fields
      Convert current status code using MEXW027_CONV for VINCENT equivalent
      Obtain Wholesale Data (join MEXW003_VEH_STATUS and MEXW007_VEH_WHS for 40V status)
      Obtain WERS vehicle line, body style, brand by querying MEXW034, MEXW032, MEXW033, MEXW031
      Get specific status dates (20T, 30R/30P, 30T, 80F) from MEXW003_VEH_STATUS
      Get last QAD wholesale dealer (40V status) from MEXW003_VEH_STATUS
      IF data successfully gathered (MEXW027-FOUND switch)
        WRITE VINCENT-RECORD from WS-VINCENT-DETAIL-RECORD
        Increment output counters
      END-IF
      Initialize WS-VINCENT-DETAIL-RECORD
    END-IF
  END-IF
END-Process_GEVIS_Vehicle_Record.
```

4.3.2 Key Algorithmic Details

- **Producer-Driven Processing:** The program iterates through a list of producers specified in the INPUT-PARM file. Each producer code is used as a DTA_DATA_SRC_C filter in DB2 queries.
- **Timestamp-Based Extraction:** The main cursors (MEXW001_CSR, MEXW003_CSR, MEXW008_CSR, SALE_CHK_CSR) use a timestamp range (WS-PREV-RUN-TIMESTAMP to WS-CURR-DRBN-TIMESTAMP) to select records updated since the last successful run. MEXS016_GENERIC2 stores this control timestamp.
- **Model Year Filtering:** Vehicle records are filtered based on model year relative to the current

year (`BDT_MDL_YR_Y BETWEEN :WS-CURR-MODEL-YY -4 AND :WS-CURR-MODEL-YY +2`). `SALE_-CHK_CSR` specifically targets older vehicles (`A.BDT_MDL_YR_Y < :WS-CURR-MODEL-YY -4`) that were sold recently.

- **WDMO Dealer Verification:** For each vehicle, subroutine `EXWWSSTK` is called to determine the current stocking dealer. The program then queries `MEXW035_DLR_MSTR` using this dealer code. If `DLR-SUB-SUBLVL1-C` from `MEXW035` is 'WDM', the vehicle is considered associated with a WDMO dealer and processed further. If `EXWWSSTK` returns SQLCODE +100 (not found), the `SEND-EMAIL` flag is set.
- **WERS Data Logic (`2160P-OBTAIN-WERS-DATA`):**
  - If data source is 'EA' or 'NA': Data is primarily fetched from `MEXW034_VL_BRAND` using `VEH-GEVIS-VL-C`.
  - For other sources:
    - If WERS string found (via `VWR-MAJ-FEAT-DFNED-F = 'Y'` in `MEXW004_VEH_WERS_-STRING`): WERS vehicle line from `MEXW004` is used. `VLN-GEVIS-VL-C` is derived or looked up in `MEXW034`.
    - Else (no WERS string): WERS vehicle line from `MEXW032_CATALOG` (via `VEH-CATALOG-C`) is used. `VLN-GEVIS-VL-C` is derived or looked up in `MEXW034`.
  - WERS Body Style (`WS-DTL-WERS-BODY-STYLE-C`) is constructed by concatenating product type and body type codes obtained from `MEXW004`, `MEXW031_CATMAP`, or `MEXW033_BODY_TYPE` based on data source and availability.
- **Status Code Conversion:** The GEVIS status code (e.g., `SSTK-CUR-STAT-C`) is converted to a VINCENT-specific status code by looking up `MEXW027_CONV` with `CNT-CND-CNV-TYP-C = 'STA-TUS'` and `CNT-DTA-DATA-SRC-C = 'VI'`. Special handling for status '800' based on sales type and date.
- **Retail Customer Data (`2137P-POPULATE-RETAIL-OUTPUT`):** If customer first and last names are blank, business names (`VRS-CST-BUS-1-N`, `VRS-CST-BUS-2-N`) are used. Sales type codes are converted via an `EVALUATE` statement in `2139P-POPULATE-CONCEPS-SLSTYP`.
- **Date Formatting:** Dates are often reformatted from YYYY-MM-DD to YYYYMMDD for output fields.
- **Error Handling and Abend:** SQL errors in critical operations trigger `9999I-ABEND`, which calls `COREDUMP`. Missing SYSPARM records also lead to an abend.

4.4 Input/Output Specifications

- **Input Files:**
  - `INPUT-PARM` (Logical Name: PARM)

    - ∗ Purpose: Contains a list of producer codes (data sources) to be processed by the program.
    - ∗ Format: Fixed length 80 bytes per record. The first 2 bytes (`INPUT-SOURCE`) are used.
    - ∗ Access Method: Sequential read.

- **Output Files:**
  - ○ `AUDIT-FILE` (Logical Name: AUDIT)
    - ∗ Purpose: Records program execution details, counts of records processed, errors, and start/stop timestamps.
    - ∗ Format: Fixed length 80 bytes per record, composed of `AUDIT-LABEL` (30 bytes) and `AUDIT-DATA` (50 bytes).
    - ∗ Access Method: Sequential write (extend).
  - ○ `VINCENT-OUT` (Logical Name: VINOUT)
    - ∗ Purpose: The main output bridge file for the VINCENT system, containing extracted and formatted vehicle, customer, and dealer data.
    - ∗ Format: Fixed length 1000 bytes per record (`VINCENT-DETAIL`). Includes E&G and VINCENT specific headers and trailers.
    - ∗ Access Method: Sequential write (extend).

4.5 DB2 Database Details

The program interacts with several DB2 tables using embedded SQL statements, primarily through cursors and direct SELECT/UPDATE statements.

Cursors Declared in `WORKING-STORAGE SECTION`:

1. **`MEXW001_CSR`**
   - Purpose: Selects vehicle order details from `MEXW001_VEH_ORDER` based on data source, model year range, active status, non-blank VIN, and update timestamp.
   - SQL: `sql   DECLARE MEXW001_CSR CURSOR WITH HOLD FOR   SELECT VEH_VIN_FULL_C   ,   VEH_ORD_ID_C   ,   DTA_DATA_SRC_C   ,   BDT_MDL_YR_Y   ,   DLR_DLR_C   ,   WMI_WMI_C   ,   VEH_LCL_PLT_C   ,   VEH_LCL_BDYTYP_C   ,   VEH_GBL_ORD_DLR_C   ,   VEH_GBL_SHIP_TO_C   ,   VEH_ORD_RCPT_Y   ,   VEH_WDMO_FLEET_C   ,   VEH_WDMO_ORD_TYP   ,   VEH_CATALOG_C   ,   VEH_GBL_CATALOG_C   ,   VEH_PO_Y   ,   VEH_GEVIS_VL_C   ,   COUNTRY_ISO3_C   FROM MEXW001_VEH_ORDER   WHERE DTA_DATA_SRC_C   = :VEH-DTA-DATA-SRC-C   AND BDT_MDL_YR_Y BETWEEN :WS-CURR-MODEL-YY -4   AND :WS-CURR-MODEL-YY +2   AND VEH_ACTIVE_F   = "Y"   AND VEH_VIN_FULL_C   > " "   AND VEH_UPDT_S BETWEEN   :WS-PREV-RUN-TIMESTAMP AND   :WS-CURR-DRBN-TIMESTAMP   FOR READ ONLY`

2. **`MEXW003_CSR`**

   - Purpose: Selects vehicle order details by joining `MEXW001_VEH_ORDER` and `MEXW003_VEH_STATUS`. Filters on data source, model year, active status, VIN, status update timestamp, specific status codes, and current/active status flags.
   - SQL: `sql   DECLARE MEXW003_CSR CURSOR WITH HOLD FOR   SELECT A.VEH_VIN_FULL_C   , A.VEH_ORD_ID_C   , A.DTA_DATA_SRC_C   , A.BDT_MDL_YR_Y   , A.DLR_DLR_C   , A.WMI_WMI_C   , A.VEH_LCL_PLT_C   , A.VEH_LCL_BDYTYP_C   , A.VEH_GBL_ORD_DLR_C   , A.VEH_GBL_SHIP_TO_C   , A.VEH_ORD_RCPT_Y   , A.VEH_WDMO_FLEET_C   , A.VEH_WDMO_ORD_TYP   , A.VEH_CATALOG_C   , A.VEH_GBL_CATALOG_C   , A.VEH_PO_Y   , A.VEH_GEVIS_VL_C   , A.COUNTRY_ISO3_C   FROM MEXW001_VEH_ORDER A   , MEXW003_VEH_STATUS B   WHERE A.DTA_DATA_SRC_C   = :VEH-DTA-DATA-SRC-C   AND A.DTA_DATA_SRC_C   = B.DTA_DATA_SRC_C   AND A.VEH_ORD_ID_C   = B.VEH_ORD_ID_C   AND A.BDT_MDL_YR_Y BETWEEN :WS-CURR-MODEL-YY -4   AND :WS-CURR-MODEL-YY +2   AND A.VEH_ACTIVE_F = "Y"   AND A.VEH_VIN_FULL_C   > " "   AND B.VST_UPDT_S   BETWEEN :WS-PREV-RUN-TIMESTAMP   AND :WS-CURR-DRBN-TIMESTAMP   AND B.STA_STATUS_C   IN ("00B","30T", "40V","80J",   "80V","90U", "90V")   AND B.VST_CUR_STAT_F   = "Y"   AND B.VST_ACTIVE_F   = "Y"   FOR READ ONLY`

3. **`MEXW008_CSR`**

   - Purpose: Selects vehicle order details by joining `MEXW001_VEH_ORDER` and `MEXW008_VEH_RTL`. Filters on data source, model year, active status, VIN, retail update timestamp, and retail active flag.
   - SQL: `sql   DECLARE MEXW008_CSR CURSOR WITH HOLD FOR   SELECT A.VEH_VIN_FULL_C   , A.VEH_ORD_ID_C   , A.DTA_DATA_SRC_C   , A.BDT_MDL_YR_Y   , A.DLR_DLR_C   , A.WMI_WMI_C   , A.VEH_LCL_PLT_C   , A.VEH_LCL_BDYTYP_C   , A.VEH_GBL_ORD_DLR_C   , A.VEH_GBL_SHIP_TO_C   , A.VEH_ORD_RCPT_Y   , A.VEH_WDMO_FLEET_C   , A.VEH_WDMO_ORD_TYP   , A.VEH_CATALOG_C   , A.VEH_GBL_CATALOG_C   , A.VEH_PO_Y   , A.VEH_GEVIS_VL_C   , A.COUNTRY_ISO3_C   FROM MEXW001_VEH_ORDER A   , MEXW008_VEH_RTL   B   WHERE A.DTA_DATA_SRC_C   = :VEH-DTA-DATA-SRC-C   AND A.DTA_DATA_SRC_C   = B.DTA_DATA_SRC_C   AND A.VEH_ORD_ID_C   = B.VEH_ORD_ID_C   AND A.BDT_MDL_YR_Y BETWEEN :WS-CURR-MODEL-YY -4   AND :WS-CURR-MODEL-YY +2   AND A.VEH_ACTIVE_F   = "Y"   AND A.VEH_VIN_FULL_C   > " "   AND B.VRS_UPDT_S   BETWEEN :WS-PREV-RUN-TIMESTAMP   AND :WS-CURR-DRBN-TIMESTAMP   AND B.VRS_ACTIVE_F = "Y"   FOR READ ONLY`

4. **`SALE_CHK_CSR`**

   - Purpose: Selects older vehicles (model year < current year - 4) sold within the last 12 months by joining `MEXW001_VEH_ORDER` and `MEXW008_VEH_RTL`.

- SQL: sql DECLARE SALE_CHK_CSR CURSOR WITH HOLD FOR SELECT A.VEH_VIN_FULL_C ,A.VEH_ORD_ID_C ,A.DTA_DATA_SRC_C ,A.BDT_MDL_YR_Y ,A.WMI_WMI_C ,A.VEH_LCL_PLT_C ,A.VEH_LCL_BDYTYP_C ,A.VEH_GBL_ORD_DLR_C ,A.VEH_GBL_SHIP_TO_C ,A.VEH_ORD_RCPT_Y ,A.VEH_WDMO_FLEET_C ,A.VEH_WDMO_ORD_TYP ,A.VEH_CATALOG_C ,A.VEH_GBL_CATALOG_C ,A.VEH_PO_Y ,A.VEH_GEVIS_VL_C ,A.COUNTRY_ISO3_C FROM MEXW001_VEH_ORDER A ,MEXW008_VEH_RTL B WHERE B.DTA_DATA_SRC_C = :VRS-DTA-DATA-SRC-C AND B.VRS_UPDT_S BETWEEN :WS-PREV-RUN-TIMESTAMP AND :WS-CURR-DRBN-TIMESTAMP AND B.VRS_ACTIVE_F = "Y" AND A.BDT_MDL_YR_Y < :WS-CURR-MODEL-YY -4 AND A.VEH_VIN_FULL_C > " " AND A.VEH_ORD_ID_C = B.VEH_ORD_ID_C AND A.DTA_DATA_SRC_C = B.DTA_DATA_SRC_C FOR READ ONLY

5. `MEXW031_CSR`
   - Purpose: Selects option code and product type from `MEXW031_CATMAP` for non-'NA' and non-'EA' data sources, used to derive local body style.
   - SQL: sql DECLARE MEXW031_CSR CURSOR WITH HOLD FOR SELECT OPT_OPTION_C ,VPT_PROD_TYP_C FROM MEXW031_CATMAP WHERE DTA_DATA_SRC_C = :CTM-DTA-DATA-SRC-C AND CTM_LCL_CATALOG_C = :CTM-LCL-CATALOG-C AND OFM_OPTION_FAM_C IN ("BS", "CA") OPTIMIZE FOR 1 ROW FOR READ ONLY
6. `MEXW003_40V_CSR`
   - Purpose: Retrieves the most recent wholesale global dealer from `MEXW003_VEH_STATUS` for a '40V' status and 'WD' current data source.
   - SQL: sql DECLARE MEXW003_40V_CSR CURSOR WITH HOLD FOR SELECT VST_GBL_LOC_C FROM MEXW003_VEH_STATUS WHERE VEH_ORD_ID_C = :VST-VEH-ORD-ID-C AND DTA_DATA_SRC_C = :VST-DTA-DATA-SRC-C AND STA_STATUS_C = :VST-STA-STATUS-C AND VST_ACTIVE_F = :VST-ACTIVE-F AND VST_STAT_TYP_C = :VST-STAT-TYP-C AND VST_CUR_DATA_SRC_C = :VST-CUR-DATA-SRC-C ORDER BY VST_STAT_Y DESC ,VST_STATIC_ISRT_REC_S DESC FOR READ ONLY

Direct SQL Statements in `PROCEDURE DIVISION`:

- In `7000P-OBTAIN-DRBN-TIMESTAMP`: sql SET :WS-CURR-DRBN-TIMESTAMP = CURRENT TIMESTAMP
- In `7005P-UPDATE-TIMESTAMP` (Updates `MEXS016_GENERIC2`): sql UPDATE MEXS016_GENERIC2 SET GNT_ATTRIBUTE_DATA = :GNT-ATTRIBUTE-DATA WHERE GNT_SYSTEM_CD = :GNT-SYSTEM-CD AND GNT_TABLE_ID = :GNT-TABLE-ID AND GNT_KEY_DATA = :GNT-KEY-DATA
- In `7010P-SELECT-MEXS016` (Selects from `MEXS016_GENERIC2`): sql SELECT GNT_ATTRIBUTE_DATA INTO :GNT-ATTRIBUTE-DATA FROM MEXS016_GENERIC2 WHERE GNT_SYSTEM_CD

= :GNT-SYSTEM-CD AND GNT_TABLE_ID = :GNT-TABLE-ID AND GNT_-KEY_DATA = :GNT-KEY-DATA

- In 7020P-GET-MEXW035-DATA (Selects from MEXW035_DLR_MSTR): sql SELECT SUB_SUBLVL1_C ,DLR_SUPER_DLR_C INTO :DLR-SUB-SUBLVL1-C ,:DLR-SUPER-DLR-C FROM MEXW035_-DLR_MSTR WHERE DLR_DLR_C = :DLR-DLR-DLR-C
- In 7030P-SELECT-WERS-DATA-W004 (Selects from MEXW004_VEH_WERS_STRING): sql SELECT VWR_WERS_STRING_X ,VWR_WERS_VL_C ,VWR_WERS_PRD_TP_C ,VWR_MAJ_FEAT_DFNED_F INTO :VWR-WERS-STRING-X ,:VWR-WERS-VL-C ,:VWR-WERS-PRD-TP-C ,:VWR-MAJ-FEAT-DFNED-F FROM MEXW004_VEH_WERS_STRING WHERE VEH_ORD_ID_C = :VWR-VEH-ORD-ID-C AND DTA_DATA_SRC_C = :VWR-DTA-DATA-SRC-C
- In 7040P-SELECT-MEXW008-90V-DATA (Joins MEXW003_VEH_STATUS and MEXW008_VEH_RTL): sql SELECT A.VST_STAT_Y ,B.VRS_LCL_FLEET_C ,B.VRS_CST_FIRST_N ,B.VRS_CST_BUS_-1_N ,B.VRS_CST_BUS_2_N ,B.VRS_CST_MID_INIT_X ,B.VRS_CST_LAST_N ,B.VRS_-CST_ADDR_1_X ,B.VRS_CST_ADD_DIV2_N ,B.VRS_CST_ADD_DIV1_C ,B.VRS_CST_POSTAL_C ,B.VRS_SALESPERSON_C ,B.VRS_TYP_LCL_CUST_C ,B.VRS_RPT_SALE_Y ,B.VRS_WARR_-STRT_Y INTO :VST-STAT-Y ,:VRS-LCL-FLEET-C ,:VRS-CST-FIRST-N ,:VRS-CST-BUS-1-N ,:VRS-CST-BUS-2-N ,:VRS-CST-MID-INIT-X ,:VRS-CST-LAST-N ,:VRS-CST-ADDR-1-X ,:VRS-CST-ADD-DIV2-N ,:VRS-CST-ADD-DIV1-C ,:VRS-CST-POSTAL-C ,:VRS-SALESPERSON-C ,:VRS-TYP-LCL-CUST-C ,:VRS-RPT-SALE-Y ,:VRS-WARR-STRT-Y FROM MEXW003_VEH_STATUS A ,MEXW008_VEH_RTL B WHERE A.VEH_ORD_ID_-C = :VST-VEH-ORD-ID-C AND A.DTA_DATA_SRC_C = :VST-DTA-DATA-SRC-C AND A.STA_STATUS_C = :VST-STA-STATUS-C AND A.VST_LAST_OCCUR_F = :VST-LAST-OCCUR-F AND A.VST_ACTIVE_F = :VST-ACTIVE-F AND A.VST_AC-TIVE_F = B.VRS_ACTIVE_F AND A.STA_STATUS_C = B.STA_STATUS_C AND A.VEH_ORD_ID_C = B.VEH_ORD_ID_C AND A.DTA_DATA_SRC_C = B.DTA_DATA_-SRC_C AND SUBSTR(A.VST_LCL_LOC_C, 1,7) = B.VRS_LCL_DLR_C AND A.VST_STAT_Y = B.VRS_RETAIL_Y
- In 7045P-SELECT-MEXW027 (Selects from MEXW027_CONV): sql SELECT CNT_LCL_DATA_X INTO :CNT-LCL-DATA-X FROM MEXW027_CONV WHERE CND_CNV_TYP_C = :CNT-CND-CNV-TYP-C AND DTA_DATA_SRC_C = :CNT-DTA-DATA-SRC-C AND CNT_GBL_-DATA_X = :CNT-GBL-DATA-X
- In 7050P-SELECT-WHOLESALE (Joins MEXW003_VEH_STATUS and MEXW007_VEH_WHS): sql SE-LECT A.VST_GBL_LOC_C ,A.VST_STAT_Y ,B.VWS_TOT_LCL_A ,B.CUR_CURRENCY_C INTO :VST-GBL-LOC-C ,:VST-STAT-Y ,:VWS-TOT-LCL-A ,:VWS-CUR-CURRENCY-C FROM MEXW003_VEH_STATUS A ,MEXW007_VEH_WHS B WHERE A.VEH_ORD_ID_C = :VST-VEH-ORD-ID-C AND A.DTA_DATA_SRC_C = :VST-DTA-DATA-SRC-C AND A.STA_-STATUS_C = :VST-STA-STATUS-C AND A.VST_LAST_OCCUR_F = :VST-LAST-OCCUR-F AND A.VST_ACTIVE_F = :VST-ACTIVE-F AND A.VST_ACTIVE_F = B.VWS_ACTIVE_F AND A.STA_STATUS_C = B.STA_STATUS_C AND A.VEH_-

ORD_ID_C = B.VEH_ORD_ID_C AND A.DTA_DATA_SRC_C = B.DTA_DATA_-SRC_C AND A.VST_STAT_Y = B.VWS_DATE_Y AND SUBSTR(A.VST_LCL_LOC_C, 1,7) = B.VWS_LCL_DLR_C

- In 7060P-SELECT-MEXW034 (Selects from MEXW034_VL_BRAND): sql SELECT VLN_WERS_VL_C ,VLN_WERS_PRD_TP_C ,VLN_WERS_BRAND_C INTO :VLN-WERS-VL-C ,:VLN-WERS-PRD-TP-C ,:VLN-WERS-BRAND-C FROM MEXW034_VL_BRAND WHERE DTA_DATA_SRC_C = :VLN-DTA-DATA-SRC-C AND VLN_GEVIS_VL_C = :VLN-GEVIS-VL-C AND VLN_-ACTIVE_F = :VLN-ACTIVE-F
- In 7070P-SELECT-VL-MEXW032 (Selects from MEXW032_CATALOG): sql SELECT VHL_VEH_-LINE_C ,VPT_PROD_TYP_C INTO :CTG-VHL-VEH-LINE-C ,:CTG-VPT-PROD-TYP-C FROM MEXW032_CATALOG WHERE DTA_DATA_SRC_C = :CTG-DTA-DATA-SRC-C AND CTG_LCL_CATALOG_C = :CTG-LCL-CATALOG-C
- In 7080P-SELECT-VL-MEXW034 (Selects from MEXW034_VL_BRAND): sql SELECT VLN_GEVIS_-VL_C ,VLN_WERS_PRD_TP_C ,VLN_WERS_BRAND_C INTO :VLN-GEVIS-VL-C ,:VLN-WERS-PRD-TP-C ,:VLN-WERS-BRAND-C FROM MEXW034_VL_BRAND WHERE DTA_DATA_-SRC_C = :VLN-DTA-DATA-SRC-C AND VLN_WERS_VL_C = :VLN-WERS-VL-C AND VLN_ACTIVE_F = :VLN-ACTIVE-F AND VLN_WERS_PRD_TP_C = :VLN-WERS-PRD-TP-C
- In 7085P-SELECT-MEXW033 (Selects from MEXW033_BODY_TYPE): sql SELECT BDT_WERS_BDY_-TYP_C INTO :BDT-WERS-BDY-TYP-C FROM MEXW033_BODY_TYPE WHERE BDT_PROD_SRC_-C = :BDT-PROD-SRC-C AND BDT_BDY_TYP_C = :BDT-BDY-TYP-C AND BDT_START_YR_R <= :BDT-START-YR-R AND BDT_END_YR_R >= :BDT-END-YR-R
- In 7090P-SELECT-MEXW003 (Selects from MEXW003_VEH_STATUS): sql SELECT VST_STAT_-Y INTO :VST-STAT-Y FROM MEXW003_VEH_STATUS WHERE VEH_ORD_ID_C = :WS-VST-VEH-ORD-ID-C AND DTA_DATA_SRC_C = :WS-VST-DTA-DATA-SRC-C AND STA_STATUS_C = :WS-VST-STA-STATUS-C AND VST_LAST_OCCUR_F = :WS-VST-LAST-OCCUR-F AND VST_ACTIVE_F = :WS-VST-ACTIVE-F

Tables Referenced:

- MEXW001_VEH_ORDER
- MEXW003_VEH_STATUS
- MEXW008_VEH_RTL
- MEXS016_GENERIC2
- MEXW031_CATMAP
- MEXW035_DLR_MSTR
- MEXW004_VEH_WERS_STRING

- `MEXW027_CONV`
- `MEXW007_VEH_WHS`
- `MEXW034_VL_BRAND`
- `MEXW032_CATALOG`
- `MEXW033_BODY_TYPE`

(Table `MEXW021_SUBLVL_ASG` is included via `COPY CPEWD021` but does not appear to be directly referenced by SQL in the main program EXWWB909. It might be used by the subprogram `EXWWSSTK` or was intended for future use.)

4.6 IMS Database Details

No IMS databases are referenced in the program.

4.7 Called Sub-routine/Program Details

- `EXWWSSTK`
  - Purpose: To obtain the current stocking dealer, current stocking dealer status code, and current stocking dealer status date for a given vehicle order.
  - Called from: `2110P-CALL-EXWWSSTK`.
  - Parameters: `SSTK-I-O-DATA` (defined in `CPEWSSTK` copybook).
    - Input: `SSTK-MODE` ('I' for inquiry), `SSTK-DTA-DATA-SRC-C`, `SSTK-VEH-ORD-ID-C`.
    - Output: `SSTK-GBL-STK-DLR-C`, `SSTK-LCL-STK-DLR-C`, `SSTK-STK-DLR-STAT-C`, `SSTK-STK-DLR-STAT-Y`, `SSTK-CUR-STAT-C`, `SSTK-CUR-STAT-Y`, etc. Also returns error details in `SSTK-OUT-DATA-MSG`.
- `COREDUMP`
  - Purpose: To generate a system dump for abend diagnosis.
  - Called from: `2115P-SSTK-FATAL-ERROR` (if `SSTK-SQL-RETURN-CODE` is not +100 and not 0), `9999I-ABEND`.
  - Parameters: None explicitly shown.

4.8 VSAM File Details

No VSAM files are referenced in the program.

4.9 IBM MQ Details

No IBM MQ series details are referenced in the program.

4.10 CICS Details

No CICS details are referenced in the program. It is a batch program.

4.11 Error Handling

- **Paragraph Name**: `0700P-GET-SYSPARM-RECORD`
  - **Trigger Condition(s):**
    - `INPUT-PARM` read `AT END` and `WS-NBR-SYSPARM-RECS-READ <= ZERO`.
  - **Action Taken:**
    - Moves "MISSING SYSPARM RECORDS" to `ABEND-MSG`.
    - Moves "PARAGRAPH 0700P" to `ABEND-MSG-2`.
    - Performs `9999I-ABEND`.
  - **Status Codes / Messages / Variables affected:**
    - `ABEND-MSG`, `ABEND-MSG-2`.
- **Paragraph Name**: `2110P-CALL-EXWWSSTK`
  - **Trigger Condition(s):**
    - `SSTK-DB2-ERROR` is true after calling `EXWWSSTK`.
  - **Action Taken:**
    - Performs `2115P-SSTK-FATAL-ERROR`.
  - **Status Codes / Messages / Variables affected:**
    - None directly, delegates to `2115P`.
- **Paragraph Name**: `2115P-SSTK-FATAL-ERROR`
  - **Trigger Condition(s):**
    - Called when `EXWWSSTK` returns `SSTK-DB2-ERROR`.
    - `SSTK-SQL-RETURN-CODE = 100` (Not Found from `EXWWSSTK`).
    - `SSTK-SQL-RETURN-CODE` is any other non-zero, non-100 error.
  - **Action Taken:**
    - Writes extensive `EXWWSSTK` error details to `AUDIT-FILE`.
    - If `SSTK-SQL-RETURN-CODE = 100`: Sets `SEND-EMAIL` to `TRUE`, increments `WS-NBR-EXWWSSTK-NOTFOUND-CALLS`.
    - If other SQL error: Calls `COREDUMP`.

- **Status Codes / Messages / Variables affected:**
  - `AUDIT-RECORD` (multiple writes).
  - `SEND-EMAIL` switch.
  - `WS-NBR-EXWWSSTK-NOTFOUND-CALLS` counter.
  - Potential program abend via `COREDUMP`.

- **Paragraph Name**: `2117P-MISSING-MEXW035-ROW`
  - **Trigger Condition(s):**
    - `MEXW035-NOT-FOUND` is true after `7020P-GET-MEXW035-DATA` (dealer not found in master).
  - **Action Taken:**
    - Writes "MISSING DEALER ON MEXW035", dealer code, and VIN to `AUDIT-FILE`.
    - Increments `WS-NBR-MEXW035-NOTFOUND-CALLS`.
  - **Status Codes / Messages / Variables affected:**
    - `AUDIT-RECORD`.
    - `WS-NBR-MEXW035-NOTFOUND-CALLS`.

- **Paragraph Name**: `2145P-GET-MEXW027-INFO`
  - **Trigger Condition(s):**
    - `MEXW027-NOT-FOUND` is true after `7045P-SELECT-MEXW027` (status conversion not found).
  - **Action Taken:**
    - Writes "MISSING STATUS ON MEXW027", status code, and VIN to `AUDIT-FILE`.
    - Increments `WS-NBR-MEXW027-NOTFOUND-CALLS`.
  - **Status Codes / Messages / Variables affected:**
    - `AUDIT-RECORD`.
    - `WS-NBR-MEXW027-NOTFOUND-CALLS`.

- **Paragraph Name**: General SQL Error Handling (e.g., `7000P`, `7005P`, `7010P`, `7020P`, `7030P`, `7045P`, `7050P`, `7060P`, `7070P`, `7080P`, `7085P`, `7090P`, `8000P` to `8507P` series)
  - **Trigger Condition(s):**
    - `SQLCODE` is not `SC-DB2-SQLCODE-OK` (+0) or `SC-DB2-SQLCODE-NOT-FOUND` (+100) (or `SC-DB2-SQLCODE-INVALID-DATE` for `8000P-OPEN-MEXW001-CSR` which is handled as fatal).
  - **Action Taken:**

    - For SELECTs/OPENs: Writes relevant key/parameter data to `AUDIT-FILE`.
    - Moves `SQLCODE` to `DB2-ABEND-SQLCODE`.
    - Sets `DB2-ABEND-FUNCTION` (e.g., "SELECT", "OPEN", "FETCH", "CLOSE", "UPDATE").
    - Sets `DB2-ABEND-TABLE` (e.g., "MEXS016_GENERIC2", "MEXW001_CSR").
    - Moves `DB2-ABEND-MSG` to `ABEND-MSG`.
    - Sets `ABEND-PARAGRAPH` to the current paragraph name.
    - Performs `9999I-ABEND`.
  - **Status Codes / Messages / Variables affected:**
    - `SQLCODE`, `SC-DB2-SQLCODE`.
    - `AUDIT-RECORD`.
    - `DB2-ABEND-MSG`, `ABEND-MSG`, `ABEND-PARAGRAPH`.

- **Paragraph Name**: `9999I-ABEND`
  - **Trigger Condition(s):**
    - Performed by other paragraphs upon detecting a fatal error.
  - **Action Taken:**
    - Writes `ABEND-MSG` and `ABEND-MSG-2` to `AUDIT-FILE`.
    - Calls `COREDUMP`.
  - **Status Codes / Messages / Variables affected:**
    - `AUDIT-RECORD`.
    - Program terminates via `COREDUMP`.

5. Interface Design

5.1 External Interfaces

- **GEVIS System (DB2 Tables):**
  - EXWWB909 reads data from multiple GEVIS DB2 tables (e.g., `MEXW001_VEH_ORDER`, `MEXW003_VEH_STATUS`, `MEXW008_VEH_RTL`, `MEXW035_DLR_MSTR`, etc.) to extract vehicle, customer, and dealer information.
  - It updates the `MEXS016_GENERIC2` table with run control information (last run timestamp, batch number).
- **VINCENT System:**
  - EXWWB909 produces an outbound bridge file (`VINCENT-OUT`) which is consumed by the VINCENT system. This file contains the extracted and processed data for VINCENT's re-review processes.
- **Subprogram `EXWWSSTK`:**
  - An external COBOL subprogram called to retrieve current stocking dealer information for a vehicle. Interface is via `CALL 'EXWWSSTK' USING SSTK-I-O-DATA`.
- **Subprogram `COREDUMP`:**
  - An external utility called to force a system dump in case of unrecoverable errors.
- **SYSPARM File (`INPUT-PARM`):**
  - An input file that provides a list of producer codes, dictating which data sources the program should process.
- **Email Notification (Implicit):**
  - If `EXWWSSTK` returns a +100 SQLCODE (dealer not found), the program sets `WS-SEND-EMAIL` to 'Y', and `RETURN-CODE` to 3. This suggests an external JCL step or system monitoring tool may use this return code to trigger an email to the helpdesk.

5.2 User Interface

This program is a batch program and does not have a direct user interface. Interactions are through the input SYSPARM file and output files (audit report and VINCENT bridge file).

6. Testing Strategy

6.1 Test Plan

A comprehensive test plan should cover:

- **Unit Testing:**
  - Test individual paragraph logic, especially data extraction from each cursor (`1100P` to `1400P`) and data aggregation in `2120P-PROCESS-GEVIS-DETAIL-REC`.
  - Verify WDMO dealer logic in `2105P-VERIFY-WDMO-DEALER`, including `EXWWSSTK` call.
  - Test WERS data retrieval logic in `2160P` and its sub-paragraphs.
  - Test status code conversions in `2145P-GET-MEXW027-INFO`.
  - Validate calculations for batch numbers and timestamp handling.
- **Integration Testing:**
  - Test interaction with `EXWWSSTK` subprogram, mocking various return codes.
  - Verify correct reading from `INPUT-PARM` and processing of multiple producers.
  - Ensure accurate reads from and updates to `MEXS016_GENERIC2`.
  - Test DB2 interactions with all cursors and direct SQL, using test data that covers various scenarios (data found, data not found, multiple rows, boundary conditions for model years and timestamps).
- **System Testing:**
  - Process a full cycle with sample `INPUT-PARM` and DB2 data.
  - Validate the format and content of `VINCENT-OUT` (headers, details, trailers).
  - Verify the `AUDIT-FILE` for correct headers, trailers, and processing counts.
  - Test error handling: SQL errors, `EXWWSSTK` errors, missing SYSPARM, missing data in lookup tables (`MEXW035`, `MEXW027`).
  - Verify correct `RETURN-CODE` setting for email notification.
- **Regression Testing:**
  - Execute existing test cases after any modifications to ensure no unintended side effects, especially considering its clones (EXWWB910, EXWWGEVP).

6.2 Testing Environment

- A dedicated test environment mirroring production as closely as possible.
- Test DB2 database with controlled datasets for `MEXW001_VEH_ORDER`, `MEXW003_VEH_STATUS`, `MEXW008_VEH_RTL`, `MEXS016_GENERIC2`, and all other lookup tables.

- Test versions of `EXWWSSTK` and `COREDUMP` (or stubs for `COREDUMP`).
- Sample `INPUT-PARM` files covering various producer scenarios.
- JCL to execute the batch program.
- Tools to inspect output files (`VINCENT-OUT`, `AUDIT-FILE`) and DB2 table contents.

7. Performance Considerations

7.1 Performance Analysis

- The program processes data based on SYSPARM input, iterating through producers. For each producer, multiple DB2 cursors are opened and processed.
- `MEXW001_CSR`, `MEXW003_CSR`, `MEXW008_CSR`, and `SALE_CHK_CSR` are the main driving cursors. Their performance is critical. They filter on `DTA_DATA_SRC_C`, `BDT_MDL_YR_Y` (range), and various update timestamps (`VEH_UPDT_S`, `VST_UPDT_S`, `VRS_UPDT_S`). Indexes on these columns, particularly composite indexes, are crucial.
- The `BETWEEN` clauses for model year and timestamps can be performance-intensive if not properly indexed.
- Joins in `MEXW003_CSR`, `MEXW008_CSR`, `SALE_CHK_CSR`, `7040P-SELECT-MEXW008-90V-DATA`, and `7050P-SELECT-WHOLESALE` require efficient join strategies from DB2. Key columns used in joins (`VEH_ORD_ID_C`, `DTA_DATA_SRC_C`, `STA_STATUS_C`, `VST_STAT_Y`) should be indexed.
- The `SUBSTR(A.VST_LCL_LOC_C, 1,7) = B.VRS_LCL_DLR_C` (and similar in `7050P`) might make index usage difficult on `VST_LCL_LOC_C`.
- Frequent single-row lookups (e.g., `7020P-GET-MEXW035-DATA`, `7045P-SELECT-MEXW027`, `7060P-SELECT-MEXW034`, etc.) should be fast if primary keys or well-defined indexes are used. `MEXW031_CSR` uses `OPTIMIZE FOR 1 ROW`.
- The call to `EXWWSSTK` for each vehicle record selected from the main cursors can be a bottleneck if `EXWWSSTK` itself is not performant or involves complex DB2 lookups.
- File I/O to `AUDIT-FILE` and `VINCENT-OUT` is sequential and should generally be efficient, but large volumes of output data can impact overall runtime.

7.2 Optimization Recommendations

- **DB2 Indexing:**
  - Ensure optimal indexes exist for all tables, especially on columns used in `WHERE` clauses (particularly `DTA_DATA_SRC_C`, `BDT_MDL_YR_Y`, `VEH_ACTIVE_F`, `VEH_VIN_FULL_C`, various timestamp columns like `VEH_UPDT_S`, `VST_UPDT_S`, `VRS_UPDT_S`) and join conditions (`VEH_ORD_ID_C`).
  - Consider composite indexes that match the order of predicates in `WHERE` clauses.
  - Review `EXPLAIN` plans for all SQL statements to ensure efficient access paths are being used.
- **SQL Query Tuning:**
  - For `SUBSTR` in join conditions, investigate if this can be redesigned or if a function-based index (if supported and appropriate) could help.

  - Ensure statistics are up-to-date for all referenced DB2 tables.

- **Subprogram `EXWWSSTK`:** Analyze the performance of `EXWWSSTK`. If it's slow, optimize it, as it's called frequently.
- **Data Volume:** If the volume of data processed per producer or overall is very large, consider if processing can be parallelized by producer, or if commit frequency (if applicable, though this program seems to be read-heavy except for `MEXS016` update) needs adjustment.
- **Working Storage Management:** Ensure efficient handling of large data structures like `WS-VINCENT-DETAIL-RECORD` if many records are held in memory before I/O (though this program writes record by record).
- **Cursor Optimization:** The cursors are declared `WITH HOLD`. This is appropriate if commits are issued during cursor processing and the cursor needs to remain open. However, if no commits are happening within the loop, `WITH HOLD` might not be strictly necessary and can have minor overhead. In this program, `MEXS016` updates occur outside the main cursor loops, so `WITH HOLD` is likely for maintaining position across SYSPARM record processing if it were designed for commits between producers (not apparent here).

8. Appendices

8.1 Glossary

- **GEVIS:** (Acronym not fully defined in context) - A system owning and maintaining the source DB2 tables.
- **VINCENT:** North American Incentive Claiming System.
- **VRULES:** (Acronym not fully defined in context) - Programs within VINCENT for claim validation.
- **WDMO:** (Acronym not fully defined in context) - A classification for dealers; records associated with WDMO dealers are prioritized.
- **SYSPARM:** System parameter file, used here to input producer codes.
- **E&G:** (Acronym not fully defined in context) - Likely refers to a standard header/trailer format.
- **CSR:** Cursor (e.g., `MEXW001_CSR`).
- **DCLGEN:** Declaration Generator - A utility that creates COBOL record layouts from DB2 table definitions.
- **WERS:** Worldwide Engineering Release System.
- **QAD:** (Acronym not fully defined in context) - Likely another system or data source qualifier, e.g., "FTC QAD Current Data Source (WD)".

8.2 References

- **Source Code:** EXWWB909.cbl
- **Copybooks:**
  - CPEWD001 (MEXW001_VEH_ORDER DCLGEN)
  - CPEWD003 (MEXW003_VEH_STATUS DCLGEN)
  - CPEWD004 (MEXW004_VEH_WERS_STRING DCLGEN)
  - CPEWD007 (MEXW007_VEH_WHS DCLGEN)
  - CPEWD008 (MEXW008_VEH_RTL DCLGEN)
  - CPESD016 (MEXS016_GENERIC2 DCLGEN)
  - CPEWD021 (MEXW021_SUBLVL_ASG DCLGEN)
  - CPEWD027 (MEXW027_CONV DCLGEN)
  - CPEWD031 (MEXW031_CATMAP DCLGEN)
  - CPEWD032 (MEXW032_CATALOG DCLGEN)

  - CPEWD033 (MEXW033_BODY_TABLE DCLGEN)
  - CPEWD034 (MEXW034_VL_BRAND DCLGEN)
  - CPEWD035 (MEXW035_DLR_MSTR DCLGEN)
  - CPESGNTB (Generic Table Layouts - EXSE System)
  - CPEWGNTB (Generic Table Layout - EXWW System)
  - CPESEBWS (BMPSHELL Working Storage)
  - CPEWSSTK (Parameters for EXWWSSTK subprogram)
  - CPESDB2 (SQLCA and SQLCODES)
  - CPEWVNCT (VINCENT Header/Trailer Layout)
- **Called Subprograms:**
  - EXWWSSTK
  - COREDUMP

End of COBOL Technical Design Specification for Modernization