COBOL - Technical Design Specification for Modernization: EXWWB910

# 1. Introduction

## 1.1 Purpose

The EXWWB910 program is a batch COBOL application designed to extract vehicle, customer, and dealer data from various DB2 tables, which are owned and maintained by the GEVIS system. Its primary purpose is to create an outbound bridge file. This bridge file is subsequently used by VINCENT (North American Incentive Claiming System) to re-review all VRULES programs and compare them to all VIN records. VINCENT utilizes this data for chargeback (automatic post-claim validation) and auto claim generation, particularly when a “full pass” in VRULES is requested by the business.

## 1.2 Scope

The scope of EXWWB910 includes: \* Reading an input SYSPARM file containing a list of producers. \* Processing one producer at a time. \* For each producer, opening a DB2 cursor on MEXW001\_VEH\_ORDER to select vehicle orders based on model year criteria (currently 4 years back and 2 years forward from the current run year, plus specific sold vehicle criteria). \* Filtering records to process only those associated with a current WDMO (Wholesale Distribution Management Operations) dealer. \* Extracting data from various GEVIS DB2 tables including MEXW001\_VEH\_ORDER, MEXW003\_VEH\_STATUS, MEXW004\_VEH\_WERS\_STRING, MEXW007\_VEH\_WHS, MEXW008\_VEH\_RTL, MEXW027\_CONV, MEXW031\_CATMAP, MEXW032\_CATALOG, MEXW033\_BODY\_TYPE, MEXW034\_VL\_BRAND, and MEXW035\_DLR\_MSTR. \* Populating an outbound bridge file with the extracted data, including standard E&G (Enterprise & GEVIS) headers/trailers and VINCENT specific headers/trailers. \* Handling cases where data fields from DB2 tables might be unpopulated by moving spaces to alphanumeric fields and zeros to numeric fields in the output. \* Performing IMS checkpointing for restart capabilities. \* Generating an audit report with processing statistics. \* Sending an email notification to the help desk under certain error conditions (e.g., missing data in MEXW027\_CONV).

## 1.3 Audience

This Technical Design Specification (TDD) is intended for: \* COBOL Developers and System Analysts responsible for maintaining, modifying, or modernizing the EXWWB910 program. \* Database Administrators who manage the DB2 tables accessed by the program. \* System Architects involved in understanding the program’s role within the GEVIS and VINCENT ecosystems. \* Testing and Quality Assurance teams responsible for validating the program’s functionality. \* Operations personnel responsible for running and monitoring the batch job.

# 2. Overview

## 2.1 Background

EXWWB910 was originally created in July 2006 to facilitate data exchange between the GEVIS and VINCENT systems. It serves as a crucial data extraction tool, providing VINCENT with comprehensive vehicle, customer, and dealer information necessary for incentive claiming and validation processes. Over the years, the program has undergone numerous revisions to accommodate changing business requirements, data source modifications, performance enhancements, and error handling improvements. These include adjustments to model year selection criteria, changes in how dealer information is retrieved, integration of WERS string data, and implementation of checkpointing.

## 2.2 Objectives

The primary objectives of EXWWB910 are: \* To accurately extract specified vehicle, customer, and dealer data from GEVIS DB2 tables. \* To transform and format the extracted data according to the layout required by the VINCENT system. \* To produce a reliable outbound bridge file for consumption by VINCENT. \* To ensure data integrity by correctly handling missing or unpopulated data fields. \* To process data efficiently for the specified producers. \* To provide audit trails and error reporting for operational monitoring and issue resolution.

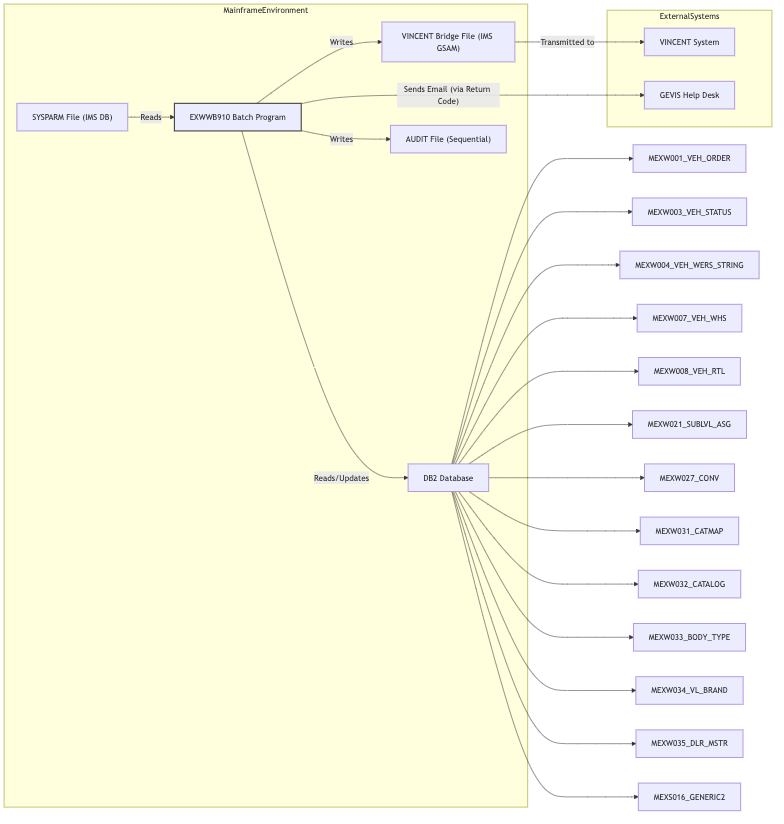
## 2.3 Assumptions and Constraints

* The program operates in a batch environment.
* DB2 and IMS services must be available and operational during program execution.
* The input SYSPARM file is correctly formatted and available.
* The DB2 tables (MEXW001, MEXW003, etc.) are populated and accessible with the required data.
* The copybooks defining data structures and DB2/IMS interfaces are accurate and up-to-date.
* Called sub-programs (CBLTDLI, COREDUMP) are available in the system’s load library.
* The program relies on specific data values (e.g., status codes, source codes) in DB2 tables for its logic.
* Network connectivity for database access is stable.
* The program processes producers sequentially as listed in the SYSPARM file.
* The program handles potential SQL +100 (NOT FOUND) conditions by specific logic or error reporting, and other SQL errors typically lead to an abend.

# 3. System Architecture

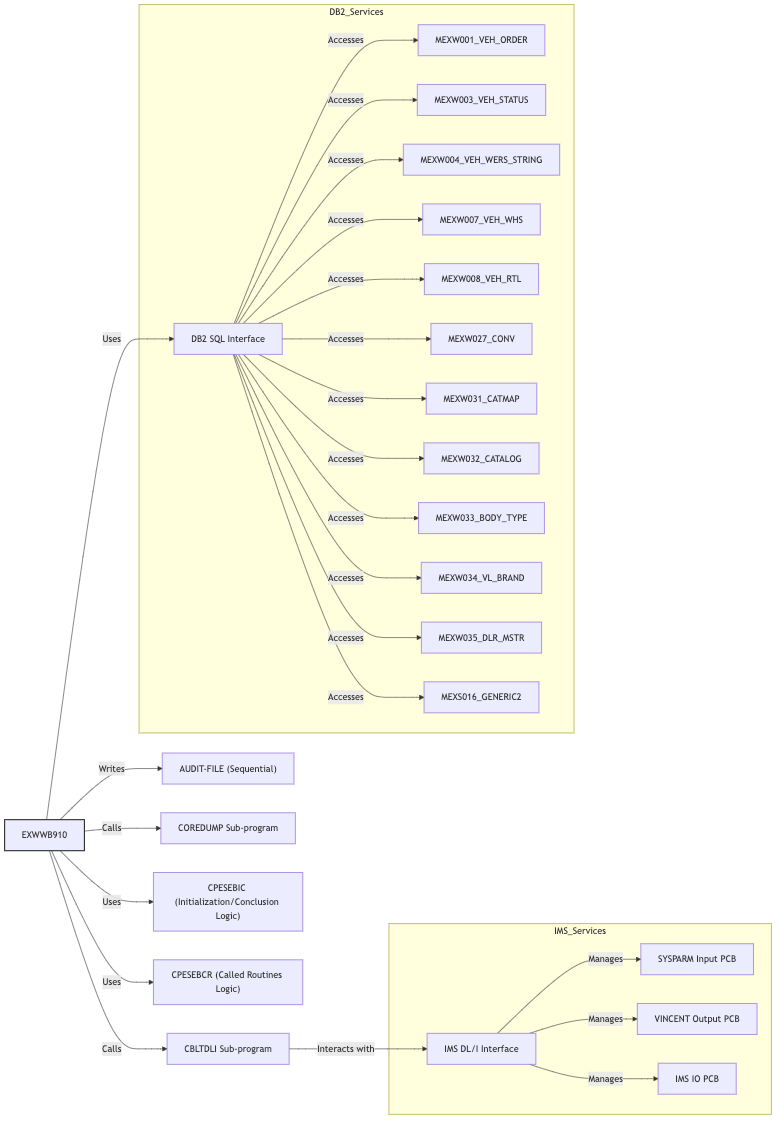
## 3.1 System Context Diagram

### 3.1 System Context Diagram



3.1 System Context Diagram

### 3.2 Component Diagram



3.2 Component Diagram

# 4. Detailed Design  
## 4.1 Program Structure  
The program EXWWB910 follows a standard batch processing structure, organized into several phases:  
  
1. \*\*Initialization (0000P-MAINLINE, 0100I-INITIALIZATION, 0400P-INITIALIZE-OTHER)\*\*:  
 \* Opens the AUDIT-FILE.  
 \* Retrieves current date/time and compile date/time. Writes initial audit headers.  
 \* Handles IMS checkpoint/restart logic using `CPESEBIC` routines (`0110I-GET-CHECKPOINT-PARM`, `9600I-IMS-RESTART`, `9500I-IMS-CHECKPOINT`).  
 \* Initializes working storage variables, switches, and obtains the current Dearborn timestamp (`7000C-OBTAIN-DRBN-TIMESTAMP`).  
 \* Calculates current model year and batch number by reading/updating `MEXS016\_GENERIC2` (`7300C-GET-BATCH-NBR`).  
 \* Reads the first SYSPARM record to get the initial producer (`8000C-GET-SYSPARM-RECORD`). Abends if SYSPARM is missing.  
 \* Writes HUB Header and VINCENT Header records to the output file (`7400C-POPULATE-HUB-HEADER`, `7500C-POPULATE-VINCENT-HEADER`, `6000C-WRITE-VINCENT-RECORD`).  
 \* Calculates a timestamp for one year prior for sales check cursor.  
  
2. \*\*Main Processing Loop (1000P-PROCESS)\*\*:  
 \* This loop iterates for each producer read from the SYSPARM file until `END-OF-SYSPARM-FILE` is true.  
 \* \*\*Process Current Model Year Vehicles (MEXW001\_CSR related logic)\*\*:  
 \* Opens `MEXW001\_CSR` to select vehicles for the current producer based on model year (current year -4 to +2) and active status (`2010C-OPEN-MEXW001-CSR`).  
 \* Fetches records from `MEXW001\_CSR` one by one (`2020C-FETCH-MEXW001-CSR`).  
 \* For each vehicle fetched, performs `2000C-PROCESS-GEVIS-VEHICLE`.  
 \* Closes `MEXW001\_CSR` after all records for the current producer are processed (`2030C-CLOSE-MEXW001-CSR`).  
 \* \*\*Process Older Sold Vehicles (SALE\_CHK\_CSR related logic)\*\*:  
 \* Opens `SALE\_CHK\_CSR` to select vehicles sold in the past year that are older than model year (current year -4) (`2100C-OPEN-SALE-CHK-CSR`).  
 \* Fetches records from `SALE\_CHK\_CSR` one by one (`2120C-FETCH-SALE-CHK-CSR`).  
 \* For each vehicle fetched, performs `2040C-PROCESS-SALE-CHK-CSR`.  
 \* Closes `SALE\_CHK\_CSR` after all records are processed (`2140C-CLOSE-SALE-CHK-CSR`).  
 \* Reads the next SYSPARM record for the next producer (`8000C-GET-SYSPARM-RECORD`).  
  
3. \*\*Vehicle Detail Processing (2000C-PROCESS-GEVIS-VEHICLE, 2040C-PROCESS-SALE-CHK-CSR, 5000C-PROCESS-GEVIS-DETAIL-REC)\*\*:  
 \* Called for each vehicle record retrieved by the main cursors.  
 \* Initializes WERS string found switch and `MEXW004-VEH-WERS-STRING`.  
 \* Selects WERS data from `MEXW004\_VEH\_WERS\_STRING` (`7600C-SELECT-WERS-DATA-W004`).  
 \* Moves data from the main cursor record (MEXW001 or SALE\_CHK) to the detail output record structure (`5020C-MOVE-MEXW001-TO-DTL`). This includes retrieving super dealer code from `MEXW035\_DLR\_MSTR` (`5050C-GET-MEXW035-DATA`) and determining body style (from WERS string or `MEXW031\_CATMAP` via `5300C/5320C/5340C`).  
 \* Obtains retail sales data by joining `MEXW003\_VEH\_STATUS` and `MEXW008\_VEH\_RTL` (`5060C-OBTAIN-RETAIL-DATA`, `5065C-SELECT-MEXW008-90V-DATA`, `5070C-POPULATE-RETAIL-OUTPUT`). This includes converting customer type codes (`5075C-POPULATE-CONCEPS-SLSTYP`).  
 \* Populates current stocking dealer information, current status, and converts status code using `MEXW027\_CONV` (`5040C-MOVE-CURR-STOCK-TO-DTL`, `7700C-SELECT-CURR-STAT-W003`, `5045C-SELECT-MEXW027-DATA`). Handles missing `MEXW027` rows (`9100C-MISSING-MEXW027-ROW`).  
 \* Obtains wholesale data by joining `MEXW003\_VEH\_STATUS` and `MEXW007\_VEH\_WHS` (`5080C-OBTAIN-WHOLESALE-DATA`, `5085C-SELECT-MEXW003-40V`).  
 \* Obtains WERS vehicle line, brand, and body style from `MEXW034\_VL\_BRAND`, `MEXW032\_CATALOG`, or `MEXW033\_BODY\_TYPE` based on data source and WERS string availability (`5100C-OBTAIN-WERS-DATA` and its sub-paragraphs `5102C`, `5104C`, `5110C`, `5112C`, `5114C`, `5115C`).  
 \* Obtains various status dates (Scheduled Target, Produced, Released, Arrived) from `MEXW003\_VEH\_STATUS` for specific status codes (`5120C-OBTAIN-MEXW003-20T`, `5140C-OBTAIN-MEXW003-30R`, `5160C-OBTAIN-MEXW003-30P`, `5180C-OBTAIN-MEXW003-30T`, `5200C-OBTAIN-MEXW003-80F`).  
 \* Obtains the most recent wholesale global dealer from `MEXW003\_VEH\_STATUS` using `MEXW003\_40V\_CSR` (`5220C-OPEN-40V-CSR`, `5230C-FETCH-40V-ROW`, `5240C-CLOSE-40V-CSR`).  
 \* If all necessary data is found (especially `MEXW027\_FOUND`), increments counters, populates HUB record ID/type/sequence, moves the detail record to the output record, and writes it (`6000C-WRITE-VINCENT-RECORD`).  
 \* Performs IMS checkpoint if frequency is met (`9400I-INCREMENT-CHKP-COUNT` from `CPESEBCR`).  
 \* Initializes the detail record for the next vehicle.  
  
4. \*\*Conclusion (0000P-MAINLINE, 0200I-CONCLUSION, 0700P-CONCLUDE-OTHER)\*\*:  
 \* Updates the run timestamp and batch number in `MEXS016\_GENERIC2` (`7250C-UPDATE-TIMESTAMP`, `7350C-UPDATE-BATCH-NBR`).  
 \* Writes VINCENT Trailer and HUB Trailer records to the output file (`7550C-POPULATE-VINCENT-TRAILER`, `7450C-POPULATE-HUB-TRAILER`, `6000C-WRITE-VINCENT-RECORD`).  
 \* Writes final audit statistics (`6020C-WRITE-AUDIT-DETAIL`).  
 \* Writes program end messages to audit report and closes AUDIT-FILE using `CPESEBIC` routines (`9000I-GET-CURRENT-DATE-TIME`, `9120I-WRITE-AUDIT-TRAILER`).  
 \* Sets RETURN-CODE to 3 if an email needs to be sent (e.g., due to missing `MEXW027` data).  
 \* Ends program execution.  
  
5. \*\*Error Handling and Abend (Throughout, 9999I-ABEND, 9999C-CALL-COREDUMP)\*\*:  
 \* Handles DB2 SQL errors and IMS status errors by moving error details to `ABEND-MSG` and performing `9999I-ABEND`.  
 \* `9999I-ABEND` (from `CPESEBCR`) writes abend messages to `AUDIT-FILE`, performs an IMS ROLB call via `CBLTDLI`, and then calls `COREDUMP`.  
 \* Specific error conditions like missing `MEXW035` or `MEXW027` data trigger audit messages and may set flags for email notification (`9000C-MISSING-MEXW035-ROW`, `9100C-MISSING-MEXW027-ROW`).  
  
## 4.2 Data Structures  
  
\*\*File Description Records:\*\*  
  
\* \*\*AUDIT-RECORD\*\*  
 \* Purpose: Defines the layout for records written to the AUDIT-FILE, used for logging processing statistics and error messages.  
 \* Record Layout:  
 ```COBOL  
 01 AUDIT-RECORD.  
 05 AUDIT-LABEL PIC X(30).  
 05 AUDIT-DATA PIC X(50).  
 ```  
 \* Copybooks Referenced: None directly for this FD, but fields like `AUDIT-LABEL-MSG` and `AUDIT-DATA-MSG` are used to populate it.  
  
\*\*Working-Storage Interface Records:\*\*  
  
\* \*\*SYSPARM-RECORD\*\*  
 \* Purpose: Defines the layout of the input SYSPARM record, read via IMS PCB, which contains the producer data source code.  
 \* Record Layout:  
 ```COBOL  
 01 SYSPARM-RECORD.  
 05 SYSPARM-DATA-SRC PIC X(02).  
 05 FILLER PIC X(78).  
 ```  
 \* Copybooks Referenced: None.  
  
\* \*\*WS-VINCENT-OUTPUT-RECORD\*\*  
 \* Purpose: A 1000-byte general-purpose record area used to build and write various record types (Headers, Trailers, Detail) to the VINCENT output GSAM file via IMS PCB.  
 \* Record Layout:  
 ```COBOL  
 01 WS-VINCENT-OUTPUT-RECORD PIC X(1000).  
 ```  
 \* Populated by: `HUB-HEADER`, `HUB-TRAILER` (from `CPEWHUB`), `VINCENT-HEADER`, `VINCENT-TRAILER` (from `CPEWVNCT`), and `WS-VINCENT-DETAIL-RECORD`.  
 \* Copybooks Referenced: None directly, but populated using layouts from `CPEWHUB` and `CPEWVNCT`.  
  
\* \*\*HUB-HEADER\*\* (from `CPEWHUB`)  
 \* Purpose: Defines the E&G HUB Header record written to the VINCENT output file.  
 \* Record Layout:  
 ```COBOL  
 05 HUB-HEADER.  
 10 HUB-HDR-RECORD-TYPE PIC X(07) VALUE " HEADER".  
 10 FILLER PIC X(01) VALUE SPACES.  
 10 HUB-HDR-ENTITY-CODE PIC X(05).  
 10 FILLER PIC X(01) VALUE SPACES.  
 10 HUB-HDR-LAYOUT-ID PIC X(20).  
 10 FILLER PIC X(01) VALUE SPACES.  
 10 HUB-HDR-TIMESTAMP PIC X(26).  
 10 FILLER PIC X(02) VALUE SPACES.  
 10 HUB-HDR-BATCH-NBR PIC 9(10).  
 10 FILLER PIC X(01) VALUE SPACES.  
 10 HUB-HDR-PROCESS-ID PIC X(20).  
 10 FILLER PIC X(01) VALUE SPACES.  
 10 HUB-HDR-TOTAL-REC-CNT PIC 9(07).  
 10 FILLER PIC X(22) VALUE SPACES.  
 10 HUB-HDR-PRIMARY-CONTACT PIC X(60).  
 10 FILLER PIC X(01) VALUE SPACES.  
 10 HUB-HDR-SECOND-CONTACT PIC X(60).  
 10 FILLER PIC X(755) VALUE SPACES.  
 ```  
 \* Copybooks Referenced: `CPEWHUB`.  
  
\* \*\*HUB-TRAILER\*\* (from `CPEWHUB`)  
 \* Purpose: Defines the E&G HUB Trailer record written to the VINCENT output file.  
 \* Record Layout:  
 ```COBOL  
 05 HUB-TRAILER.  
 10 HUB-TRL-RECORD-TYPE PIC X(07) VALUE "9TRAILR".  
 10 FILLER PIC X(01) VALUE SPACES.  
 10 HUB-TRL-ENTITY-CODE PIC X(05).  
 10 FILLER PIC X(01) VALUE SPACES.  
 10 HUB-TRL-LAYOUT-ID PIC X(20).  
 10 FILLER PIC X(01) VALUE SPACES.  
 10 HUB-TRL-TIMESTAMP PIC X(26).  
 10 FILLER PIC X(02) VALUE SPACES.  
 10 HUB-TRL-BATCH-NBR PIC 9(10).  
 10 FILLER PIC X(01) VALUE SPACES.  
 10 HUB-TRL-PROCESS-ID PIC X(20).  
 10 FILLER PIC X(01) VALUE SPACES.  
 10 HUB-TRL-TOTAL-REC-CNT PIC 9(07).  
 10 FILLER PIC X(22) VALUE SPACES.  
 10 HUB-TRL-PRIMARY-CONTACT PIC X(60).  
 10 FILLER PIC X(01) VALUE SPACES.  
 10 HUB-TRL-SECOND-CONTACT PIC X(60).  
 10 FILLER PIC X(755) VALUE SPACES.  
 ```  
 \* Copybooks Referenced: `CPEWHUB`.  
  
\* \*\*VINCENT-HEADER\*\* (from `CPEWVNCT`)  
 \* Purpose: Defines the VINCENT specific Header record written to the VINCENT output file.  
 \* Record Layout:  
 ```COBOL  
 05 VINCENT-HEADER.  
 10 VNT-HDR-HUB-LINE-NBR PIC 9(06).  
 10 VNT-HDR-HUB-REC-ID PIC X(25).  
 10 VNT-HDR-HUB-REC-ID-NBR REDEFINES  
 VNT-HDR-HUB-REC-ID.  
 15 VNT-HDR-REC-ID-ZEROES PIC 9(16).  
 15 VNT-HDR-REC-ID-NBR PIC 9(09).  
 10 VNT-HDR-HUB-REC-TYPE PIC X(03) VALUE "001".  
 10 VNT-HDR-HUB-SEQ-NBR PIC X(03) VALUE "001".  
 10 VNT-HDR-ID PIC X(04) VALUE "1HDR".  
 10 VNT-HDR-REC-TYPE PIC X(08).  
 10 VNT-HDR-CURR-BATCH-NBR PIC 9(05).  
 10 VNT-HDR-PREV-BATCH-NBR PIC 9(05).  
 10 VNT-HDR-LOW-VALUES PIC X(02) VALUE LOW-VALUES.  
 10 VNT-HDR-CURR-DATE PIC 9(08).  
 10 VNT-HDR-CURR-TIME PIC 9(06).  
 10 FILLER PIC X(925) VALUE SPACES.  
 ```  
 \* Copybooks Referenced: `CPEWVNCT`.  
  
\* \*\*VINCENT-TRAILER\*\* (from `CPEWVNCT`)  
 \* Purpose: Defines the VINCENT specific Trailer record written to the VINCENT output file.  
 \* Record Layout:  
 ```COBOL  
 05 VINCENT-TRAILER.  
 10 VNT-TRL-HUB-LINE-NBR PIC 9(06).  
 10 VNT-TRL-HUB-REC-ID PIC X(25).  
 10 VNT-TRL-HUB-REC-ID-NBR REDEFINES  
 VNT-TRL-HUB-REC-ID.  
 15 VNT-TRL-REC-ID-ZEROES PIC 9(16).  
 15 VNT-TRL-REC-ID-NBR PIC 9(09).  
 10 VNT-TRL-HUB-REC-TYPE PIC X(03) VALUE "999".  
 10 VNT-TRL-HUB-SEQ-NBR PIC X(03) VALUE "001".  
 10 VNT-TRL-ID PIC X(04) VALUE "9TRL".  
 10 VNT-TRL-REC-TYPE PIC X(08).  
 10 VNT-TRL-CURR-BATCH-NBR PIC 9(05).  
 10 FILLER PIC X(05) VALUE SPACES.  
 10 VNT-TRL-COUNTER PIC 9(07).  
 10 VNT-TRL-HIGH-VALUES PIC X(02) VALUE HIGH-VALUES.  
 10 FILLER PIC X(932) VALUE SPACES.  
 ```  
 \* Copybooks Referenced: `CPEWVNCT`.  
  
\* \*\*WS-VINCENT-DETAIL-RECORD\*\*  
 \* Purpose: Defines the detailed data record for a vehicle, customer, and dealer, written to the VINCENT output file.  
 \* Record Layout (Hierarchical Outline):  
 ```  
 01 WS-VINCENT-DETAIL-RECORD.  
 05 WS-DTL-HUB-LINE-NBR PIC 9(06).  
 05 WS-DTL-HUB-REC-ID PIC X(25).  
 05 WS-DTL-HUB-REC-ID-NBR REDEFINES WS-DTL-HUB-REC-ID.  
 10 WS-DTL-REC-ID-ZEROS PIC 9(16).  
 10 WS-DTL-REC-ID-NBR PIC 9(09).  
 05 WS-DTL-HUB-REC-TYPE PIC X(03).  
 05 WS-DTL-HUB-REC-SEQ-NBR PIC 9(03).  
 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(590).  
 05 INPUT-OUTPUT-PARAMETERS PIC X(4020).  
 ```  
 \* Copybooks Referenced: None directly, but fields are populated from various DB2 table copybooks (e.g., `CPEWD001`, `CPEWD003`, `CPEWD008`).  
  
\* \*\*CHKP-SAVE-AREA\*\*  
 \* Purpose: Used for IMS checkpoint and restart. This area is passed to `CBLTDLI` during `CHKP` and `XRST` calls to save and restore program state.  
 \* Record Layout (Hierarchical Outline - includes direct definitions and items from copied members):  
 ```  
 01 CHKP-SAVE-AREA.  
 05 LIT-00A PIC X(03) VALUE "00A".  
 ... (all other 05-level literals LIT-xxx)  
 05 LIT-TBL-VEH-ORDER PIC X(18) VALUE "MEXW001\_VEH\_ORDER ".  
 ... (all other 05-level table name literals LIT-TBL-xxx)  
 05 LIT-SALE-CHK-CSR PIC X(18) VALUE "SALE\_CHK\_CSR".  
 ... (all other 05-level miscellaneous literals)  
 05 PL-PGM-NAME PIC X(08) VALUE "EXWWB910".  
 ... (other PL-xxx literals)  
 05 WS-LAYOUT-ID PIC X(20) VALUE "WGEVVNTEXTOUT010".  
 ... (other WS-xxx constant literals)  
 05 WS-VEH-VIN-BLANK PIC X(17) VALUE " ".  
 05 WS-ORDER-ID-BLANK PIC X(25) VALUE " ".  
 05 WS-SWITCHES.  
 10 WS-MEXW001-SW PIC X(01).  
 88 MEXW001-FOUND VALUE "Y".  
 88 MEXW001-NOT-FOUND VALUE "N".  
 ... (all other 10-level switches and their 88 levels)  
 05 WS-VARIABLES.  
 10 WS-NBR-ROWS-READ-MEXW001 PIC S9(9) COMP.  
 ... (all other 10-level numeric counters WS-NBR-xxx)  
 10 WS-PREV-RUN-TIMESTAMP PIC X(26).  
 10 WS-PREV-BATCH-NBR PIC 9(05).  
 10 WS-CURRENT-BATCH-NBR PIC 9(05).  
 10 WS-CURR-DRBN-TIMESTAMP PIC X(26).  
 10 WS-CURR-DRBN-TIMESTAMP-REDEF REDEFINES WS-CURR-DRBN-TIMESTAMP.  
 15 WS-DRBN-YEAR PIC 9(04).  
 ... (redefinition sub-fields)  
 10 WS-CURR-DRBN-TMSTMP-LESS-1 PIC X(26).  
 10 WS-CURR-DRBN-TMSTMP-LESS-1-R REDEFINES WS-CURR-DRBN-TMSTMP-LESS-1.  
 15 WS-DRBN-YEAR-LESS-1 PIC 9(04).  
 ... (redefinition sub-fields)  
 10 WS-VEH-BDT-MDL-YR-Y PIC 9(04).99.  
 10 WS-VEH-BDT-MDL-YR-Y-RDF REDEFINES WS-VEH-BDT-MDL-YR-Y.  
 ... (redefinition sub-fields)  
 10 WS-CURR-MODEL-YY PIC S9(04)V99 COMP-3.  
 10 WS-VST-GBL-LOC-C-REFORMAT PIC 9(06).  
 10 WS-DLR-SUPER-DLR-C-REFORMAT PIC 9(06).  
 10 WS-DTL-VEH-GBL-DLR-C-REFORMAT PIC 9(06).  
 10 WS-PROCESS-SQLCODE-EDIT PIC +++++.  
 10 AUDIT-RECORD-MSG.  
 15 AUDIT-LABEL-MSG PIC X(10).  
 15 AUDIT-DATA-MSG PIC X(70).  
 05 WS-VALID-VARIABLES.  
 10 WS-WERS-VL-C PIC X(02).  
 88 WS-VALID-VEH-LINE-CDS VALUE "KI", "K5", "DB".  
 \* Items from COPY CPEWHUB.  
 05 HUB-HEADER. (Layout as described above)  
 05 HUB-TRAILER. (Layout as described above)  
 \* Items from COPY CPEWVNCT.  
 05 VINCENT-HEADER. (Layout as described above)  
 05 VINCENT-TRAILER. (Layout as described above)  
 ```  
 \* Copybooks Referenced: `CPEWHUB`, `CPEWVNCT`. (These are copied \*into\* the `CHKP-SAVE-AREA` 01 level).  
  
\*\*Linkage Section Records (IMS PCBs):\*\*  
  
\* \*\*IO-PCB\*\*  
 \* Purpose: Standard IMS I/O Program Communication Block used for checkpoint calls and general IMS communication.  
 \* Record Layout:  
 ```COBOL  
 01 IO-PCB.  
 05 IO-PCB-LTERM PIC X(08).  
 05 FILLER PIC X(02).  
 05 IO-PCB-STATUS PIC X(02).  
 05 FILLER PIC X(28).  
 ```  
 \* Copybooks Referenced: Implicitly through IMS interface.  
  
\* \*\*SYSPARM-PCB\*\*  
 \* Purpose: IMS PCB for accessing the input SYSPARM file/database which provides producer codes.  
 \* Record Layout:  
 ```COBOL  
 01 SYSPARM-PCB.  
 05 SYSPARM-PCB-NAME PIC X(08).  
 05 FILLER PIC X(02).  
 05 SYSPARM-PCB-STATUS PIC X(02).  
 05 FILLER PIC X(28).  
 ```  
 \* Copybooks Referenced: Implicitly through IMS interface.  
  
\* \*\*VINCENT-PCB\*\*  
 \* Purpose: IMS PCB for writing records to the VINCENT output GSAM file.  
 \* Record Layout:  
 ```COBOL  
 01 VINCENT-PCB.  
 05 VINCENT-PCB-NAME PIC X(08).  
 05 FILLER PIC X(02).  
 05 VINCENT-PCB-STATUS PIC X(02).  
 05 FILLER PIC X(28).  
 ```  
 \* Copybooks Referenced: Implicitly through IMS interface.  
  
## 4.3 Algorithms  
### 4.3.1 Overall Program Logic (Condensed Pseudocode)

PERFORM INITIALIZATION Open AUDIT-FILE Get current timestamp, compile timestamp Write audit headers Initialize checkpoint/restart (get parms, XRST, CHKP) Initialize working storage, get run timestamp, calculate model year Get/Update batch number from/to MEXS016\_GENERIC2 Read first SYSPARM record (producer) IF SYSPARM empty THEN ABEND Write HUB Header to VINCENT output Write VINCENT Header to VINCENT output Set timestamp for 1 year ago

PERFORM MAIN-PROCESS UNTIL no more SYSPARM records Open MEXW001\_CSR for current producer Fetch first record from MEXW001\_CSR PERFORM PROCESS-GEVIS-VEHICLE (from MEXW001\_CSR) UNTIL MEXW001\_CSR is empty Close MEXW001\_CSR

Open SALE\_CHK\_CSR for current producer Fetch first record from SALE\_CHK\_CSR PERFORM PROCESS-SALE-CHK-VEHICLE (from SALE\_CHK\_CSR) UNTIL SALE\_CHK\_CSR is empty Close SALE\_CHK\_CSR

Read next SYSPARM record (producer) END-PERFORM

PERFORM CONCLUSION Update run timestamp in MEXS016\_GENERIC2 Update batch number in MEXS016\_GENERIC2 Write VINCENT Trailer to VINCENT output Write HUB Trailer to VINCENT output Write audit statistics Write audit trailer, close AUDIT-FILE Set return code for email if needed GOBACK

PROCESS-GEVIS-VEHICLE / PROCESS-SALE-CHK-VEHICLE: Initialize detail record, WERS string data SELECT WERS data from MEXW004 Move base vehicle data (from cursor) to output detail record SELECT super dealer from MEXW035 for ordering dealer IF WERS string found THEN use it for body style ELSE SELECT body style from MEXW031 (if not NA/EA) or use MEXW001 body style Obtain RETAIL data (join MEXW003 and MEXW008) Populate customer info, sales dates Convert customer type Populate current stocking dealer info SELECT super dealer from MEXW035 for current dealer SELECT current status from MEXW003 SELECT VINCENT status code from MEXW027 using current GEVIS status Handle missing MEXW027 (write audit, set email flag) Adjust status code (600 or 801) based on sales type/date or customer type Obtain WHOLESALE data (join MEXW003 and MEXW007) Populate invoice dealer, invoice date, total amount (USD only) Obtain WERS details (vehicle line, brand, body style) Logic based on data source (NA/EA vs. others) and WERS string availability Access MEXW034, MEXW032, MEXW033 as needed Obtain various status dates (20T, 30R/30P, 30T, 80F) from MEXW003 Obtain last QAD wholesale global dealer from MEXW003 (40V cursor) IF required data is present (MEXW027\_FOUND) THEN Increment counters Populate HUB record ID details Move detail record to VINCENT output record Write record to VINCENT output file Perform checkpoint if count reached Initialize detail record for next vehicle

### 4.3.2 Key Algorithmic Details  
\* \*\*Producer Processing\*\*: The program reads producers one by one from an IMS SYSPARM file. For each producer, it processes two main sets of vehicles.  
\* \*\*Main Vehicle Selection (MEXW001\_CSR)\*\*:  
 \* Selects from `MEXW001\_VEH\_ORDER` joined with `MEXW035\_DLR\_MSTR`.  
 \* Criteria: `DTA\_DATA\_SRC\_C` from SYSPARM, `BDT\_MDL\_YR\_Y` between (Current Year - 4) and (Current Year + 2), active vehicle (`VEH\_ACTIVE\_F = 'Y'`), `DIV\_DIV\_C = 'EX'`, `SUB\_SUBLVL1\_C = 'WDM'`.  
 \* Ordered by `VEH\_ORD\_ID\_C`, `DTA\_DATA\_SRC\_C`.  
\* \*\*Sold Vehicle Selection (SALE\_CHK\_CSR)\*\*:  
 \* Selects from `MEXW001\_VEH\_ORDER` joined with `MEXW008\_VEH\_RTL` and `MEXW035\_DLR\_MSTR`.  
 \* Criteria: `VRS.DTA\_DATA\_SRC\_C` from SYSPARM, `VRS\_UPDT\_S` > (Current Timestamp - 1 year), `VRS\_ACTIVE\_F = 'Y'`, `VEH.BDT\_MDL\_YR\_Y` < (Current Year - 4), `DIV\_DIV\_C = 'EX'`, `SUB\_SUBLVL1\_C = 'WDM'`.  
\* \*\*WDMO Dealer Check\*\*: Only vehicles associated with a current WDMO dealer are processed further. This check involves `MEXW035\_DLR\_MSTR` where `SUB\_SUBLVL1\_C = 'WDM'`.  
\* \*\*Data Extraction from Multiple Tables\*\*: For each selected vehicle, data is gathered from:  
 \* `MEXW004\_VEH\_WERS\_STRING`: WERS string data.  
 \* `MEXW008\_VEH\_RTL` (joined with `MEXW003`): Retail customer and sales data (status '90V').  
 \* `MEXW007\_VEH\_WHS` (joined with `MEXW003`): Wholesale data (status '40V').  
 \* `MEXW003\_VEH\_STATUS`: Various status dates ('20T', '30P', '30R', '30T', '80F'), current status, current stocking dealer info, last QAD wholesale dealer.  
 \* `MEXW035\_DLR\_MSTR`: Super dealer codes for ordering and current stocking dealers.  
 \* `MEXW027\_CONV`: To convert GEVIS status codes to VINCENT status codes.  
 \* `MEXW031\_CATMAP`, `MEXW032\_CATALOG`, `MEXW033\_BODY\_TYPE`, `MEXW034\_VL\_BRAND`: For WERS vehicle line, brand, and body style information, with logic depending on data source ('NA'/'EA' vs. others) and WERS string availability.  
\* \*\*Status Code Logic\*\*:  
 \* If `WS-DTL-CUR-STA-STATUS-C` (from `MEXW027`) is '800':  
 \* If customer type `VRS-TYP-LCL-CUST-C` is "A4" (dealership use), status becomes '801'.  
 \* Else if sales type or sales receipt date is blank, status becomes '600'.  
\* \*\*Super Dealer Code\*\*: Global ordering dealer and current global stocking dealer are populated with super dealer codes from `MEXW035\_DLR\_MSTR`. If no super dealer code, spaces are moved.  
\* \*\*Date Formatting\*\*: Dates from DB2 (YYYY-MM-DD) are reformatted to YYYYMMDD for the output file.  
\* \*\*Checkpointing\*\*: IMS checkpoints are taken via `CBLTDLI` call with `SL-FUNC-CHKP` after a certain number of records are written (frequency from `MEXS016\_GENERIC2` via `BMPCHKP` table ID). `CPESEBIC` and `CPESEBCR` handle the checkpoint logic.  
\* \*\*Header/Trailer Generation\*\*: Standard E&G HUB headers/trailers and VINCENT specific headers/trailers are written to the output file at the beginning and end of processing. Batch numbers and timestamps are included.  
  
## 4.4 Input/Output Specifications  
\* \*\*Input Files:\*\*  
 \* \*\*SYSPARM File (IMS Database/Segment)\*\*:  
 \* Accessed via `SYSPARM-PCB`.  
 \* Read using `GN` (Get Next) calls to `CBLTDLI`.  
 \* Provides a list of producer codes (`SYSPARM-DATA-SRC`) to be processed.  
 \* Record Layout: `SYSPARM-RECORD` (see section 4.2).  
 \* \*\*DB2 Tables\*\*: Various GEVIS and system tables (see section 4.5 for details).  
\* \*\*Output Files:\*\*  
 \* \*\*VINCENT Bridge File (IMS GSAM File)\*\*:  
 \* Accessed via `VINCENT-PCB`.  
 \* Written using `ISRT` (Insert) calls to `CBLTDLI`.  
 \* Contains extracted and formatted vehicle, customer, and dealer data.  
 \* Record Layout: `WS-VINCENT-OUTPUT-RECORD` (1000 bytes), populated with `HUB-HEADER`, `VINCENT-HEADER`, `WS-VINCENT-DETAIL-RECORD`, `VINCENT-TRAILER`, `HUB-TRAILER` (see section 4.2).  
 \* \*\*AUDIT-FILE (Sequential File)\*\*:  
 \* `SELECT AUDIT-FILE ASSIGN TO AUDIT.`  
 \* Opened in `EXTEND` mode.  
 \* Used for logging processing statistics, run start/end times, and error/abend messages.  
 \* Record Layout: `AUDIT-RECORD` (see section 4.2).  
 \* \*\*DB2 Table Updates\*\*: `MEXS016\_GENERIC2` is updated with current run timestamp and next batch number (see section 4.5).  
  
## 4.5 DB2 Database Details  
\*\*Cursors:\*\*  
  
1. \*\*MEXW001\_CSR\*\*: Main driving cursor to select vehicle orders.  
 ```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  
 ,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  
 ,VEH.COUNTRY\_ISO3\_C  
 ,VEH.DLR\_DLR\_C  
 ,DLR.DLR\_SUPER\_DLR\_C  
 FROM MEXW001\_VEH\_ORDER VEH  
 , MEXW035\_DLR\_MSTR DLR  
 WHERE VEH.DTA\_DATA\_SRC\_C = :VEH-DTA-DATA-SRC-C  
 AND VEH.VEH\_ORD\_ID\_C > :WS-ORDER-ID-BLANK  
 AND BDT\_MDL\_YR\_Y BETWEEN :WS-CURR-MODEL-YY -4  
 AND :WS-CURR-MODEL-YY +2  
 AND VEH\_ACTIVE\_F = :VEH-ACTIVE-F  
 AND VEH\_VIN\_FULL\_C > :WS-VEH-VIN-BLANK  
 AND VEH.DLR\_DLR\_C = DLR.DLR\_DLR\_C  
 AND DIV\_DIV\_C = :LIT-EX  
 AND SUB\_SUBLVL1\_C = :LIT-WDM  
 ORDER BY VEH\_ORD\_ID\_C  
 , DTA\_DATA\_SRC\_C  
 FOR READ ONLY  
 ```  
  
2. \*\*SALE\_CHK\_CSR\*\*: Selects vehicles sold in the past 12 months older than 4 model years.  
 ```SQL  
 DECLARE SALE\_CHK\_CSR CURSOR WITH HOLD FOR  
 SELECT VEH.VEH\_VIN\_FULL\_C  
 ,VEH.VEH\_ORD\_ID\_C  
 ,VEH.DTA\_DATA\_SRC\_C  
 ,VEH.BDT\_MDL\_YR\_Y  
 ,VEH.WMI\_WMI\_C  
 ,VEH.VEH\_LCL\_PLT\_C  
 ,VEH.VEH\_LCL\_BDYTYP\_C  
 ,VEH.VEH\_GBL\_ORD\_DLR\_C  
 ,VEH.VEH\_GBL\_SHIP\_TO\_C  
 ,VEH.VEH\_ORD\_RCPT\_Y  
 ,VEH.VEH\_WDMO\_FLEET\_C  
 ,VEH.VEH\_WDMO\_ORD\_TYP  
 ,VEH.VEH\_CATALOG\_C  
 ,VEH.VEH\_GBL\_CATALOG\_C  
 ,VEH.VEH\_PO\_Y  
 ,VEH.VEH\_GEVIS\_VL\_C  
 ,VEH.COUNTRY\_ISO3\_C  
 ,VEH.DLR\_DLR\_C  
 ,DLR.DLR\_SUPER\_DLR\_C  
 FROM MEXW001\_VEH\_ORDER VEH  
 ,MEXW008\_VEH\_RTL VRS  
 ,MEXW035\_DLR\_MSTR DLR  
 WHERE VRS.DTA\_DATA\_SRC\_C = :VRS-DTA-DATA-SRC-C  
 AND VRS.VRS\_UPDT\_S > :WS-CURR-DRBN-TMSTMP-LESS-1  
 AND VRS.VRS\_ACTIVE\_F = :VRS-ACTIVE-F  
 AND VEH.BDT\_MDL\_YR\_Y < :WS-CURR-MODEL-YY -4  
 AND VEH.VEH\_VIN\_FULL\_C > :WS-VEH-VIN-BLANK  
 AND VEH.VEH\_ORD\_ID\_C = VRS.VEH\_ORD\_ID\_C  
 AND VEH.DTA\_DATA\_SRC\_C = VRS.DTA\_DATA\_SRC\_C  
 AND VEH.DLR\_DLR\_C = DLR.DLR\_DLR\_C  
 AND DIV\_DIV\_C = :LIT-EX  
 AND SUB\_SUBLVL1\_C = :LIT-WDM  
 FOR READ ONLY  
 ```  
  
3. \*\*MEXW031\_CSR\*\*: Selects body style information from `MEXW031\_CATMAP` for non-'NA'/'EA' sources.  
 ```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  
 ```  
  
4. \*\*MEXW003\_40V\_CSR\*\*: Retrieves the most recent wholesale global dealer from `MEXW003\_VEH\_STATUS`.  
 ```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  
 ```  
  
\*\*Singleton SQL SELECT Statements:\*\*  
  
\* In `5045C-SELECT-MEXW027-DATA` (selects local data for conversion):  
 ```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 `5050C-GET-MEXW035-DATA` (selects dealer master data):  
 ```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 `5065C-SELECT-MEXW008-90V-DATA` (selects retail and status data):  
 ```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\_ACTIVE\_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 `5085C-SELECT-MEXW003-40V` (selects wholesale and status data):  
 ```SQL  
 SELECT 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 `5110C-SELECT-W034-DATA` (selects WERS vehicle line data):  
 ```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 `5112C-SELECT-MEXW032-WERS-VL` (selects WERS vehicle line from 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 `5114C-SELECT-MEXW034-GEVIS-VL` (selects GEVIS vehicle line data):  
 ```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 `5115C-SELECT-MEXW033-DATA` (selects WERS 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 `5120C`, `5140C`, `5160C`, `5180C`, `5200C` (select status date from `MEXW003\_VEH\_STATUS`):  
 Generic form:  
 ```SQL  
 SELECT VST\_STAT\_Y  
 INTO  
 :VST-STAT-Y  
 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 -- Specific status ('20T', '30R', etc.)  
 AND VST\_LAST\_OCCUR\_F = :VST-LAST-OCCUR-F -- 'Y'  
 AND VST\_ACTIVE\_F = :VST-ACTIVE-F -- 'Y'  
 ```  
\* In `7000C-OBTAIN-DRBN-TIMESTAMP` (get current timestamp):  
 ```SQL  
 SET :WS-CURR-DRBN-TIMESTAMP = CURRENT TIMESTAMP  
 ```  
\* In `7300C-GET-BATCH-NBR` and `9200I-SELECT-MEXS016-GENERIC2` (from `CPESEBCR`):  
 ```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 9200I, there's an additional clause: AND GNT\_SW\_ACTIVE LIKE :GNT-SW-ACTIVE  
 ```  
\* In `7600C-SELECT-WERS-DATA-W004` (selects WERS string data):  
 ```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 `7700C-SELECT-CURR-STAT-W003` (selects current status from `MEXW003`):  
 ```SQL  
 SELECT STA\_STATUS\_C  
 , VST\_STAT\_Y  
 INTO :VST-STA-STATUS-C  
 , :VST-STAT-Y  
 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 VST\_CUR\_STAT\_F = :LIT-Y  
 AND VST\_ACTIVE\_F = :LIT-Y  
 ```  
  
\*\*SQL UPDATE Statements:\*\*  
  
\* In `7250C-UPDATE-TIMESTAMP` and `7350C-UPDATE-BATCH-NBR` and `9210I-UPDATE-MEXS016-GENERIC2` (from `CPESEBCR`):  
 ```SQL  
 UPDATE MEXS016\_GENERIC2  
 SET GNT\_ATTRIBUTE\_DATA = :GNT-ATTRIBUTE-DATA -- :WS-CURR-DRBN-TIMESTAMP or :WS-CURRENT-BATCH-NBR  
 WHERE GNT\_SYSTEM\_CD = :GNT-SYSTEM-CD  
 AND GNT\_TABLE\_ID = :GNT-TABLE-ID  
 AND GNT\_KEY\_DATA = :GNT-KEY-DATA  
 ```  
  
\*\*Tables Referenced:\*\*  
\* `MEXW001\_VEH\_ORDER`  
\* `MEXW003\_VEH\_STATUS`  
\* `MEXW004\_VEH\_WERS\_STRING`  
\* `MEXW007\_VEH\_WHS`  
\* `MEXW008\_VEH\_RTL`  
\* `MEXW027\_CONV`  
\* `MEXW031\_CATMAP`  
\* `MEXW032\_CATALOG`  
\* `MEXW033\_BODY\_TYPE`  
\* `MEXW034\_VL\_BRAND`  
\* `MEXW035\_DLR\_MSTR`  
\* `MEXS016\_GENERIC2`  
  
## 4.6 IMS Database Details  
The program interacts with IMS databases/segments via `CBLTDLI` calls, using PCBs defined in the `LINKAGE SECTION`.  
  
\* \*\*IO-PCB\*\*:  
 \* Used for `CHKP` (Checkpoint) and `ROLB` (Rollback) calls, and potentially `XRST` (Extended Restart) indirectly via `CPESEBIC`.  
 \* Standard IMS I/O PCB.  
  
\* \*\*SYSPARM-PCB\*\*:  
 \* Used to read input producer codes.  
 \* Accessed in paragraph `8000C-GET-SYSPARM-RECORD`.  
 \* Function code used: `SL-FUNC-GN` (Get Next).  
 \* Segment/Data structure: `SYSPARM-RECORD`.  
  
\* \*\*VINCENT-PCB\*\*:  
 \* Used to write output records to the VINCENT bridge file (GSAM).  
 \* Accessed in paragraph `6000C-WRITE-VINCENT-RECORD`.  
 \* Function code used: `SL-FUNC-ISRT` (Insert).  
 \* Segment/Data structure: `WS-VINCENT-OUTPUT-RECORD`.  
  
## 4.7 Called Sub-routine/Program Details  
\* \*\*`CBLTDLI`\*\*  
 \* Purpose: Standard IMS-DB/DC interface module.  
 \* Called from:  
 \* `6000C-WRITE-VINCENT-RECORD` (Function: `SL-FUNC-ISRT` for VINCENT output)  
 \* `8000C-GET-SYSPARM-RECORD` (Function: `SL-FUNC-GN` for SYSPARM input)  
 \* `9500I-IMS-CHECKPOINT` in `CPESEBCR` (Function: `SL-FUNC-CHKP` for checkpoint) - called by `EXWWB910` via `PERFORM 9400I-INCREMENT-CHKP-COUNT`.  
 \* `9600I-IMS-RESTART` in `CPESEBCR` (Function: `SL-FUNC-XRST` for restart) - called by `EXWWB910` via `PERFORM 0100I-INITIALIZATION`.  
 \* `9999I-ABEND` in `CPESEBCR` (Function: `SL-FUNC-ROLB` for rollback) - called by `EXWWB910` when errors occur.  
 \* Interface: Standard DL/I call interface (function code, PCB, I/O area, etc.). Relevant parameters are defined in `CPESIMSB` and used in the calling paragraphs.  
  
\* \*\*`COREDUMP`\*\*  
 \* Purpose: To force a system dump and abend the program.  
 \* Called from:  
 \* `9999C-CALL-COREDUMP` which is called by `9999I-ABEND` (in `CPESEBCR`).  
 \* Interface: No parameters passed explicitly in the `CALL "COREDUMP"` statement visible in `EXWWB910` or `CPESEBCR`.  
  
\* \*\*Note on `CPESEBIC` and `CPESEBCR`\*\*: These are copybooks containing `PROCEDURE DIVISION` code, effectively acting as in-line subroutines rather than separately compiled and called programs. Their logic for initialization, conclusion, checkpointing, and abend handling is integrated into `EXWWB910`'s execution flow via `PERFORM` statements.  
  
## 4.8 VSAM File Details  
No VSAM files are directly referenced or accessed by the EXWWB910 program.  
  
## 4.9 IBM MQ Details  
No IBM MQ series queues are referenced or used by the EXWWB910 program.  
  
## 4.10 CICS Details  
The EXWWB910 program is a batch program and does not operate under or interact with CICS.  
  
## 4.11 Error Handling  
  
\* \*\*Paragraph Name\*\*: `0400P-INITIALIZE-OTHER`  
 \* \*\*Trigger Condition(s):\*\*  
 \* `END-OF-SYSPARM-FILE` is true after the first attempt to read `SYSPARM-RECORD` (i.e., SYSPARM file is empty or unreadable).  
 \* \*\*Action Taken:\*\*  
 \* Moves "MISSING SYSPARM RECORDS" to `ABEND-MSG`.  
 \* Moves "PARAGRAPH 0400P" to `ABEND-MSG-2`.  
 \* Performs `9999I-ABEND`.  
 \* \*\*Status Codes / Messages / Variables affected:\*\*  
 \* `ABEND-MSG`, `ABEND-MSG-2`. Program abends.  
  
\* \*\*Paragraph Name\*\*: `2010C-OPEN-MEXW001-CSR`  
 \* \*\*Trigger Condition(s):\*\*  
 \* `SQLCODE` is not `SC-DB2-SQLCODE-OK` after `OPEN MEXW001\_CSR`.  
 \* \*\*Action Taken:\*\*  
 \* Moves `SQLCODE` to `DB2-ABEND-SQLCODE`.  
 \* Sets `DB2-ABEND-FUNCTION` to "OPEN", `DB2-ABEND-TABLE` to `LIT-TBL-VEH-ORDER`.  
 \* Moves `DB2-ABEND-MSG` to `ABEND-MSG`.  
 \* Moves "2010C" to `ABEND-PARAGRAPH`.  
 \* Performs `9999I-ABEND`.  
 \* \*\*Status Codes / Messages / Variables affected:\*\*  
 \* `SC-DB2-SQLCODE`, `DB2-ABEND-SQLCODE`, `DB2-ABEND-FUNCTION`, `DB2-ABEND-TABLE`, `ABEND-MSG`, `ABEND-PARAGRAPH`. Program abends.  
  
\* \*\*Paragraph Name\*\*: `2020C-FETCH-MEXW001-CSR`  
 \* \*\*Trigger Condition(s):\*\*  
 \* `SQLCODE` is not `SC-DB2-SQLCODE-OK` and not `SC-DB2-SQLCODE-END-OF-CURSOR` after `FETCH MEXW001\_CSR`.  
 \* \*\*Action Taken:\*\*  
 \* Moves `SQLCODE` to `DB2-ABEND-SQLCODE`.  
 \* Sets `DB2-ABEND-FUNCTION` to "FETCH", `DB2-ABEND-TABLE` to `LIT-TBL-VEH-ORDER`.  
 \* Moves `DB2-ABEND-MSG` to `ABEND-MSG`.  
 \* Moves "2020C" to `ABEND-PARAGRAPH`.  
 \* Performs `9999I-ABEND`.  
 \* \*\*Status Codes / Messages / Variables affected:\*\*  
 \* `SC-DB2-SQLCODE`, `WS-MEXW001-SW` (set to `MEXW001-NOT-FOUND` on end-of-cursor). If other error: `DB2-ABEND-SQLCODE`, etc. Program abends.  
  
\* \*\*Paragraph Name\*\*: `2030C-CLOSE-MEXW001-CSR`  
 \* \*\*Trigger Condition(s):\*\*  
 \* `SQLCODE` is not `SC-DB2-SQLCODE-OK` after `CLOSE MEXW001\_CSR`.  
 \* \*\*Action Taken:\*\*  
 \* Moves `SQLCODE` to `DB2-ABEND-SQLCODE`.  
 \* Sets `DB2-ABEND-FUNCTION` to "CLOSE", `DB2-ABEND-TABLE` to `LIT-TBL-VEH-ORDER`.  
 \* Moves `DB2-ABEND-MSG` to `ABEND-MSG`.  
 \* Moves "2030C" to `ABEND-PARAGRAPH`.  
 \* Performs `9999I-ABEND`.  
 \* \*\*Status Codes / Messages / Variables affected:\*\*  
 \* `SC-DB2-SQLCODE`, `DB2-ABEND-SQLCODE`, etc. Program abends.  
  
\* \*\*Paragraph Name\*\*: `2100C-OPEN-SALE-CHK-CSR`  
 \* \*\*Trigger Condition(s):\*\*  
 \* `SQLCODE` is not `SC-DB2-SQLCODE-OK` after `OPEN SALE\_CHK\_CSR`.  
 \* \*\*Action Taken:\*\*  
 \* Moves `SQLCODE` to `DB2-ABEND-SQLCODE`.  
 \* Sets `DB2-ABEND-FUNCTION` to "OPEN", `DB2-ABEND-TABLE` to `LIT-SALE-CHK-CSR`.  
 \* Moves `DB2-ABEND-MSG` to `ABEND-MSG`.  
 \* Moves "2100C" to `ABEND-PARAGRAPH`.  
 \* Performs `9999I-ABEND`.  
 \* \*\*Status Codes / Messages / Variables affected:\*\*  
 \* `SC-DB2-SQLCODE`, `DB2-ABEND-SQLCODE`, etc. Program abends.  
  
\* \*\*Paragraph Name\*\*: `2120C-FETCH-SALE-CHK-CSR`  
 \* \*\*Trigger Condition(s):\*\*  
 \* `SQLCODE` is not `SC-DB2-SQLCODE-OK` and not `SC-DB2-SQLCODE-END-OF-CURSOR` after `FETCH SALE\_CHK\_CSR`.  
 \* \*\*Action Taken:\*\*  
 \* Moves `SQLCODE` to `DB2-ABEND-SQLCODE`.  
 \* Sets `DB2-ABEND-FUNCTION` to "FETCH", `DB2-ABEND-TABLE` to `LIT-SALE-CHK-CSR`.  
 \* Moves `DB2-ABEND-MSG` to `ABEND-MSG`.  
 \* Moves "2120C" to `ABEND-PARAGRAPH`.  
 \* Performs `9999I-ABEND`.  
 \* \*\*Status Codes / Messages / Variables affected:\*\*  
 \* `SC-DB2-SQLCODE`, `WS-SALE-CHK-SW` (set to `SALE-CHK-NOT-FOUND` on end-of-cursor). If other error: `DB2-ABEND-SQLCODE`, etc. Program abends.  
  
\* \*\*Paragraph Name\*\*: `2140C-CLOSE-SALE-CHK-CSR`  
 \* \*\*Trigger Condition(s):\*\*  
 \* `SQLCODE` is not `SC-DB2-SQLCODE-OK` after `CLOSE SALE\_CHK\_CSR`.  
 \* \*\*Action Taken:\*\*  
 \* Moves `SQLCODE` to `DB2-ABEND-SQLCODE`.  
 \* Sets `DB2-ABEND-FUNCTION` to "CLOSE", `DB2-ABEND-TABLE` to `LIT-SALE-CHK-CSR`.  
 \* Moves `DB2-ABEND-MSG` to `ABEND-MSG`.  
 \* Moves "2140C" to `ABEND-PARAGRAPH`.  
 \* Performs `9999I-ABEND`.  
 \* \*\*Status Codes / Messages / Variables affected:\*\*  
 \* `SC-DB2-SQLCODE`, `DB2-ABEND-SQLCODE`, etc. Program abends.  
  
\* \*\*Paragraph Name\*\*: `5040C-MOVE-CURR-STOCK-TO-DTL` (via `5045C`)  
 \* \*\*Trigger Condition(s):\*\*  
 \* `MEXW027-NOT-FOUND` is true after `PERFORM 5045C-SELECT-MEXW027-DATA`.  
 \* \*\*Action Taken:\*\*  
 \* Performs `9100C-MISSING-MEXW027-ROW`. This writes to audit, sets `SEND-EMAIL TO TRUE`, and increments `WS-NBR-MEXW027-NOTFOUND-CALLS`. Processing for this vehicle record might be skipped for output.  
 \* \*\*Status Codes / Messages / Variables affected:\*\*  
 \* `SEND-EMAIL` flag, `WS-NBR-MEXW027-NOTFOUND-CALLS`.  
  
\* \*\*Paragraph Name\*\*: `5045C-SELECT-MEXW027-DATA`  
 \* \*\*Trigger Condition(s):\*\*  
 \* `SQLCODE` from `SELECT ... FROM MEXW027\_CONV` is not `SC-DB2-SQLCODE-OK` and not `SC-DB2-SQLCODE-NOT-FOUND`.  
 \* \*\*Action Taken:\*\*  
 \* Moves `SQLCODE` to `DB2-ABEND-SQLCODE`.  
 \* Sets `DB2-ABEND-FUNCTION` to "SELECT", `DB2-ABEND-TABLE` to `LIT-TBL-CONV`.  
 \* Moves `DB2-ABEND-MSG` to `ABEND-MSG`.  
 \* Moves "5045C" to `ABEND-PARAGRAPH`.  
 \* Performs `9999I-ABEND`.  
 \* \*\*Status Codes / Messages / Variables affected:\*\*  
 \* `SC-DB2-SQLCODE`, `WS-MEXW027-SW`. If error: `DB2-ABEND-SQLCODE`, etc. Program abends.  
  
\* \*\*Paragraph Name\*\*: `5050C-GET-MEXW035-DATA`  
 \* \*\*Trigger Condition(s):\*\*  
 \* `SQLCODE` from `SELECT ... FROM MEXW035\_DLR\_MSTR` is `SC-DB2-SQLCODE-NOT-FOUND`.  
 \* \*\*Action Taken:\*\*  
 \* Sets `MEXW035-NOT-FOUND` to TRUE.  
 \* Performs `9000C-MISSING-MEXW035-ROW` (writes audit, increments counter).  
 \* \*\*Trigger Condition(s):\*\*  
 \* `SQLCODE` is not `SC-DB2-SQLCODE-OK` and not `SC-DB2-SQLCODE-NOT-FOUND`.  
 \* \*\*Action Taken:\*\*  
 \* Moves `SQLCODE` to `DB2-ABEND-SQLCODE`.  
 \* Sets `DB2-ABEND-FUNCTION` to "SELECT", `DB2-ABEND-TABLE` to `LIT-TBL-DLR-MSTR`.  
 \* Moves `DB2-ABEND-MSG` to `ABEND-MSG`.  
 \* Moves "5050C" to `ABEND-PARAGRAPH`.  
 \* Performs `9999I-ABEND`.  
 \* \*\*Status Codes / Messages / Variables affected:\*\*  
 \* `SC-DB2-SQLCODE`, `WS-MEXW035-SW`. If error: `DB2-ABEND-SQLCODE`, etc. Program abends.  
  
\* \*\*Generic SQL Error Handling (e.g., `5065C`, `5110C`, `5112C`, `5114C`, `5115C`, `5120C`, `5140C`, `5160C`, `5180C`, `5200C`, `5220C`, `5230C`, `5240C`, `5300C`, `5320C`, `5340C`, `7000C`, `7250C`, `7300C`, `7350C`, `7600C`, `7700C`)\*\*  
 \* \*\*Trigger Condition(s):\*\*  
 \* `SQLCODE` indicates an error (neither OK nor expected NOT-FOUND/END-OF-CURSOR).  
 \* \*\*Action Taken:\*\*  
 \* Moves `SQLCODE` to `DB2-ABEND-SQLCODE`.  
 \* Sets `DB2-ABEND-FUNCTION` (e.g., "SELECT", "OPEN", "FETCH", "CLOSE", "UPDATE").  
 \* Sets `DB2-ABEND-TABLE` to the relevant table literal.  
 \* Moves `DB2-ABEND-MSG` to `ABEND-MSG`.  
 \* Moves current paragraph ID to `ABEND-PARAGRAPH`.  
 \* Performs `9999I-ABEND`.  
 \* \*\*Status Codes / Messages / Variables affected:\*\*  
 \* `SC-DB2-SQLCODE`, `DB2-ABEND-SQLCODE`, etc. Program abends. In `7600C` and `7700C`, specific audit messages are written before abend for some SQL errors.  
  
\* \*\*Paragraph Name\*\*: `6000C-WRITE-VINCENT-RECORD`  
 \* \*\*Trigger Condition(s):\*\*  
 \* `VINCENT-PCB-STATUS` is not `SC-IMS-STAT-OK` after `CBLTDLI` call for `ISRT`.  
 \* \*\*Action Taken:\*\*  
 \* Moves `VINCENT-PCB-STATUS` to `IMS-ABEND-STATUS`.  
 \* Sets `IMS-ABEND-FUNCTION` to `SL-FUNC-ISRT`, `IMS-ABEND-PCB-NAME` to `VINCENT-PCB-NAME`.  
 \* Moves `IMS-ABEND-MSG` to `ABEND-MSG`.  
 \* Moves "6000C" to `ABEND-PARAGRAPH`.  
 \* Performs `9999I-ABEND`.  
 \* \*\*Status Codes / Messages / Variables affected:\*\*  
 \* `SC-IMS-STAT`, `IMS-ABEND-STATUS`, etc. Program abends.  
  
\* \*\*Paragraph Name\*\*: `8000C-GET-SYSPARM-RECORD`  
 \* \*\*Trigger Condition(s):\*\*  
 \* `SYSPARM-PCB-STATUS` is not `SC-IMS-STAT-OK` and not `SC-IMS-STAT-END-OF-DB` after `CBLTDLI` call for `GN`.  
 \* \*\*Action Taken:\*\*  
 \* Moves `SYSPARM-PCB-STATUS` to `IMS-ABEND-STATUS`.  
 \* Sets `IMS-ABEND-FUNCTION` to `SL-FUNC-GN`, `IMS-ABEND-PCB-NAME` to `SYSPARM-PCB-NAME`.  
 \* Moves `IMS-ABEND-MSG` to `ABEND-MSG`.  
 \* Moves "8000C" to `ABEND-PARAGRAPH`.  
 \* Performs `9999I-ABEND`.  
 \* \*\*Status Codes / Messages / Variables affected:\*\*  
 \* `SC-IMS-STAT`, `WS-END-OF-SYSPARM-FILE`. If error: `IMS-ABEND-STATUS`, etc. Program abends.  
  
\* \*\*Paragraph Name\*\*: `9999I-ABEND` (in `CPESEBCR`)  
 \* \*\*Trigger Condition(s):\*\*  
 \* Called by other paragraphs upon detecting unrecoverable errors.  
 \* \*\*Action Taken:\*\*  
 \* Writes `ABEND-MSG` and `ABEND-MSG-2` to `AUDIT-FILE`.  
 \* Calls `CBLTDLI` with `SL-FUNC-ROLB` to perform IMS rollback.  
 \* Calls `COREDUMP` to terminate the program and produce a dump.  
 \* \*\*Status Codes / Messages / Variables affected:\*\*  
 \* `AUDIT-FILE`. Program abends.  
  
\* \*\*Paragraph Name\*\*: `0110I-GET-CHECKPOINT-PARM` (in `CPESEBIC`)  
 \* \*\*Trigger Condition(s):\*\*  
 \* `MEXS016-NOT-FOUND` is true after `PERFORM 9200I-SELECT-MEXS016-GENERIC2` for BMPCHKP parameters.  
 \* \*\*Action Taken:\*\*  
 \* Moves "CHECKPOINT PARM NOT ACTIVE IN BMPCHKP GENERIC TABLE" to `ABEND-MSG`.  
 \* Moves "0110I" to `ABEND-PARAGRAPH`.  
 \* Performs `9999I-ABEND`.  
 \* \*\*Status Codes / Messages / Variables affected:\*\*  
 \* `ABEND-MSG`, `ABEND-PARAGRAPH`. Program abends.  
  
# 5. Interface Design  
## 5.1 External Interfaces  
\* \*\*SYSPARM File (Input)\*\*: An IMS database/segment providing producer codes that drive the main processing loop. Accessed via `SYSPARM-PCB`.  
\* \*\*GEVIS DB2 Database (Input/Update)\*\*: Numerous DB2 tables are read to extract vehicle, dealer, customer, and configuration data. `MEXS016\_GENERIC2` table is updated with run timestamp and batch number.  
\* \*\*VINCENT Bridge File (Output)\*\*: An IMS GSAM file created by the program, containing extracted and formatted data for the VINCENT system. Accessed via `VINCENT-PCB`.  
\* \*\*AUDIT File (Output)\*\*: A sequential file (`AUDIT`) used for logging run statistics, processing summaries, and error/abend information.  
\* \*\*GEVIS Help Desk Notification (Implicit Output)\*\*: If specific error conditions occur (e.g., missing `MEXW027` row), the program sets `RETURN-CODE` to 3, which is typically used by job scheduling systems to trigger email notifications.  
  
## 5.2 User Interface  
The EXWWB910 program is a batch application and does not have a direct user interface. Input is provided via the SYSPARM file, and output is generated to the VINCENT bridge file and the AUDIT report file.  
  
# 6. Testing Strategy  
## 6.1 Test Plan  
A comprehensive test plan should cover the following aspects:  
\* \*\*Unit Testing\*\*: Testing individual paragraphs and logic units, especially those involving complex data lookups (e.g., WERS data retrieval, status code conversions) and DB2/IMS interactions.  
\* \*\*Integration Testing\*\*:  
 \* Verify correct interaction with DB2 tables, ensuring data is fetched and interpreted correctly.  
 \* Verify correct interaction with IMS for SYSPARM input and VINCENT output.  
 \* Test with various producer codes in SYSPARM.  
\* \*\*System Testing\*\*:  
 \* Test with a representative set of input data covering various scenarios (e.g., vehicles with/without WERS strings, different data sources, various sales statuses, missing optional data).  
 \* Verify the correctness of the output VINCENT bridge file format and content against specifications.  
 \* Validate the HUB and VINCENT headers and trailers.  
 \* Check accuracy of audit report statistics.  
\* \*\*Error Handling Testing\*\*:  
 \* Simulate DB2 errors (e.g., table unavailability, unexpected SQLCODEs) to ensure proper abend procedures.  
 \* Simulate IMS errors (e.g., PCB status errors) for input/output operations.  
 \* Test scenarios leading to specific error messages (e.g., missing `MEXW027` or `MEXW035` data) and verify audit reporting and email notification triggers.  
\* \*\*Checkpoint/Restart Testing\*\*:  
 \* Verify that IMS checkpoints are taken correctly.  
 \* Test program restart from a checkpoint to ensure processing resumes correctly.  
\* \*\*Performance Testing\*\*: Test with large volumes of data to ensure processing completes within acceptable timeframes and resource utilization.  
\* \*\*Regression Testing\*\*: After any modification, re-run a standard set of tests to ensure existing functionality is not broken.  
  
## 6.2 Testing Environment  
The testing environment should closely mirror the production environment, including:  
\* Access to a DB2 subsystem with test versions of all GEVIS tables (`MEXW001`, `MEXW003`, etc.) populated with controlled test data.  
\* Access to an IMS environment for SYSPARM input and GSAM output.  
\* Availability of all required copybooks and called sub-programs (`CBLTDLI`, `COREDUMP`).  
\* JCL for executing the batch program.  
\* Tools for inspecting DB2 table contents, IMS queue/file contents, and sequential audit files.  
  
# 7. Appendices  
## 7.1 Glossary  
\* \*\*GEVIS\*\*: Acronym for the system owning the source DB2 tables.  
\* \*\*VINCENT\*\*: North American Incentive Claiming System, consumer of the output bridge file.  
\* \*\*VRULES\*\*: A set of validation rules within the VINCENT system.  
\* \*\*SYSPARM\*\*: An input file/parameter providing producer codes for processing.  
\* \*\*WDMO\*\*: Wholesale Distribution Management Operations. Refers to a type of dealer.  
\* \*\*DB2\*\*: IBM's relational database management system.  
\* \*\*IMS\*\*: IBM's Information Management System, a hierarchical database and transaction management system.  
\* \*\*GSAM\*\*: Generalized Sequential Access Method, used for IMS sequential file processing.  
\* \*\*PCB\*\*: Program Communication Block, used in IMS to define an application's view of a database or message queue.  
\* \*\*WERS\*\*: Worldwide Engineering Release System.  
\* \*\*VIN\*\*: Vehicle Identification Number.  
\* \*\*E&G\*\*: Enterprise & GEVIS, likely referring to a standard header/trailer format.  
\* \*\*SQLCODE\*\*: A status variable indicating the result of an SQL operation.  
\* \*\*DCLGEN\*\*: Declaration Generator, a utility to create COBOL or PL/I record descriptions from DB2 table definitions.  
  
## 7.2 References  
\* \*\*Program Source\*\*: `EXWWB910.cbl`  
\* \*\*Copybooks\*\*:  
 \* `CPEWD001` (DCLGEN for `MEXW001\_VEH\_ORDER`)  
 \* `CPEWD003` (DCLGEN for `MEXW003\_VEH\_STATUS`)  
 \* `CPEWD004` (DCLGEN for `MEXW004\_VEH\_WERS\_STRING`)  
 \* `CPEWD007` (DCLGEN for `MEXW007\_VEH\_WHS`)  
 \* `CPEWD008` (DCLGEN for `MEXW008\_VEH\_RTL`)  
 \* `CPESD016` (DCLGEN for `MEXS016\_GENERIC2`)  
 \* `CPEWD021` (DCLGEN for `MEXW021\_SUBLVL\_ASG`)  
 \* `CPEWD027` (DCLGEN for `MEXW027\_CONV`)  
 \* `CPEWD031` (DCLGEN for `MEXW031\_CATMAP`)  
 \* `CPEWD032` (DCLGEN for `MEXW032\_CATALOG`)  
 \* `CPEWD033` (DCLGEN for `MEXW033\_BODY\_TYPE`)  
 \* `CPEWD034` (DCLGEN for `MEXW034\_VL\_BRAND`)  
 \* `CPEWD035` (DCLGEN for `MEXW035\_DLR\_MSTR`)  
 \* `CPESDB2` (SQLCA and SQLCODES)  
 \* `CPESIMSB` (IMS functions and status codes)  
 \* `CPESGNTB` (Generic table layouts - EXSE System)  
 \* `CPEWGNTB` (Generic table layout - EXWW System)  
 \* `CPESEBWS` (BMPSHELL working storage)  
 \* `CPEWHUB` (E&G HUB header/trailer layout)  
 \* `CPEWVNCT` (VINCENT header/trailer layout)  
 \* `CPESEBIC` (BMPSHELL Initialization and Conclusion procedural copybook)  
 \* `CPESEBCR` (BMPSHELL Called Routines procedural copybook)  
\* \*\*Called Programs\*\*:  
 \* `CBLTDLI`  
 \* `COREDUMP`  
  
---  
End of COBOL Technical Design Specification for Modernization  
---