Technical Specification Document

# Index

1. Document Information 1

2. Introduction 1

3. Transport Management 1

4. Requirement Overview 1

5. Solution Approach 1

6. SAP Object Details 1

7. Data Declarations & SAP Tables Used 1

8. User Interface Details 1

9. Processing Logic 1

10. Detailed Logic Block Descriptions 1

11. Output Details 1

12. Enhancements & Modifications 1

13. Flow Diagram 1

14. Error Handling & Logging 1

15. Performance Considerations 1

16. Security & Authorizations 1

17. Test Scenario 1

18. Sign-Off 1

# 1. Document Information

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Document Title | Project Name | SAP System/Release Version | Client Name | Prepared By (Author, Department) | Document Version & Date |
| ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 | ECC |  | PWC AI Asset | 1.0 / 2024-06-10 |

# 2. Introduction

This document provides the technical specification for the SAP ABAP program ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 (type: PROG). The objective of this document is to formally describe the program’s purpose, logic, and structure, including its key subroutines, data declarations, and selection screen elements. The scope covers all major components of the program, such as internal table definitions, class and method implementations, and the sequence of data retrieval and processing steps performed via subroutines. This specification is intended for SAP ABAP developers, technical consultants, and reviewers who require a detailed understanding of the program’s design and functionality for purposes of development, maintenance, or code review.

# 3. Transport Management

[Error: Section 'Transport Management' generation failed after 3 retries.]

# 4. Requirement Overview

The business requirement is to develop an ABAP program (ZRCOPY\_SAMPLE\_ECC\_CODE\_V1) that enables users to efficiently retrieve, process, and prepare key business data from SAP ECC tables for reporting or further processing. The solution provides a user-friendly selection screen where users can specify a plant and a range of material numbers to filter the data. Upon execution, the program systematically fetches and validates plant and material master data, retrieves delivery items, pricing conditions, billing documents, financial postings, customer and branch information, and material storage details from various standard SAP tables such as MARA, LIPS, T001W, VBRK, VBRP, ACDOCA, KNA1, and others. The program processes and consolidates this data into internal tables, applies necessary business logic, and prepares a final dataset for output. The objective is to streamline data extraction and preparation processes, reduce manual effort, and ensure accurate, consistent information is available for business analysis and decision-making. This solution is designed to be modular, maintainable, and easily extensible to accommodate future business needs.

# 5. Solution Approach

1. The solution leverages a modular and layered ABAP architecture, utilizing a main program (ZRCOPY\_SAMPLE\_ECC\_CODE\_V1) with multiple INCLUDE files to separate concerns such as data declarations, selection screen definitions, and subroutine implementations, ensuring maintainability and reusability.

2. Data processing is orchestrated through a series of well-defined subroutines (FORM routines) that encapsulate specific business logic for data retrieval, transformation, and validation from core SAP tables (e.g., MARA, LIPS, VBRK, VBRP, ACDOCA), as well as custom or auxiliary tables, supporting a clear and traceable workflow.

3. The approach combines procedural and object-oriented ABAP principles, with internal tables and structures for bulk data handling, selection screens for dynamic user input, and local class definitions for encapsulating data and methods, enabling flexible, scalable, and robust data processing aligned with SAP best practices.

# 6. SAP Object Details

|  |  |  |  |
| --- | --- | --- | --- |
| Object Type | Object Name | Description | Related Main Program/Module |
| Program | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 | Main ABAP executable program for data retrieval and processing using subroutines and includes. | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| Include | ZRCOPY\_SAMPLE\_ECC\_CODE\_F01\_V1 | Include file containing subroutine (FORM) implementations and class/method definitions. | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| Include | ZRCOPY\_SAMPLE\_ECC\_CODE\_SEL\_V1 | Include file containing selection screen definitions (PARAMETERS, SELECT-OPTIONS). | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| Include | ZRCOPY\_SAMPLE\_ECC\_CODE\_TOP\_V1 | Include file containing TABLES statements and global data declarations. | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| Class | lcl\_data | Local class definition for encapsulating data and methods (e.g., billing document data). | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| Method | get\_data | Method of class lcl\_data; clears the billing document variable gv\_vbrk. | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_and\_check\_plant | Subroutine to check if plant parameter is provided and fetch plant data from T001W. | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_material | Subroutine to fetch material data from MARA table based on selected material numbers. | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_delivery\_items | Subroutine to fetch delivery item data from LIPS table based on material and plant conditions. | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | prepare\_final\_data | Subroutine to process delivery items and populate the final output internal table. | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_konv | Subroutine to fetch condition records (knumv) from PRCD\_ELEMENTS table into internal table. | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | populate\_salary | Subroutine to extract monetary value from ACDOCA table and store in a local variable. | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetc\_vbrk | Subroutine to fetch a billing document number (vbeln) from VBRK table where draft status is empty. | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_vbrp | Subroutine to fetch billing document item data (vbeln, posnr) from VBRP table where draft status is empty. | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_bsak | Subroutine to fetch financial document data from ACDOCA table (company code, year, doc number, etc.). | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_j1m0cust | Subroutine to fetch customer numbers from KNA1 table. | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_marc\_stawn | Subroutine to fetch MARC table fields and call methods of /SAPSLL/CL\_MM\_CLS\_SERVICE for commodity code classification. | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_dzaehk | Subroutine to fetch a single condition counter from PRCD\_ELEMENTS table. | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_jbbranch | Subroutine to fetch company code and business place data from P\_BusinessPlace table. | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_marc\_mard | Subroutine to fetch material numbers from MARC and storage location data from MARD table. | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_vbuk | Subroutine to count the number of entries in VBAK table (sales document headers). | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_orderby | Subroutine to fetch and process material data from MARA table with dynamic substring and ordering logic. | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_single | Subroutine to fetch a single material number from MARC table based on a substring of another material number. | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| Class | /SAPSLL/CL\_MM\_CLS\_SERVICE | SAP standard class for commodity code classification and details (used in fetch\_marc\_stawn). | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| Method | get\_commodity\_code\_cls | Method of /SAPSLL/CL\_MM\_CLS\_SERVICE; retrieves commodity code classification. | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| Method | get\_commodity\_code\_details | Method of /SAPSLL/CL\_MM\_CLS\_SERVICE; retrieves commodity code details. | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| Selection Screen | PARAMETERS p\_werks | Selection screen parameter for plant (T001W-WERKS). | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| Selection Screen | SELECT-OPTIONS s\_matnr | Selection screen select-option for material number (MARA-MATNR). | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |

# 7. Data Declarations & SAP Tables Used

|  |  |  |  |
| --- | --- | --- | --- |
| Declaration Name | Data Type/Object | Description | Usage Context |
| ikonv | Internal Table (with header line) of structure KONV | Internal table for storing condition records with KONV structure | Used to store and process pricing condition records, populated in 'fetch\_konv' subroutine |
| ifinal | Internal Table (with header line) with fields vbeln, posnr, matnr, werks | Custom internal table for final output with sales document, item, material, and plant fields | Populated in 'prepare\_final\_data' subroutine as the final result set |
| i\_t001w | Internal Table (with header line) of type T001W | Internal table for plant master data | Used in 'fetch\_and\_check\_plant' to store plant data |
| t001w | t001w | Work area for T001W table | Used for accessing plant master data |
| mara | mara | Work area for MARA table | Used for accessing material master data |
| lips | lips | Work area for LIPS table | Used for accessing delivery item data |
| bseg | bseg | Work area for BSEG table | Used for accessing accounting document segment data |
| ilips | OCCURS 0 structure based on LIPS | Internal table for delivery items with LIPS structure | Populated in 'fetch\_delivery\_items' subroutine |
| imara | OCCURS 0 structure based on MARA | Internal table for materials with MARA structure | Populated in 'fetch\_material' subroutine |
| lv\_product | TYPE matnr | Variable for storing a product (material) number | Used for temporary storage of material number |
| lv\_salary | TYPE dmbtr | Variable for storing a salary amount | Used for temporary storage of monetary values, e.g., in 'populate\_salary' |
| zrcopy\_sample\_ecc\_code\_top\_v1 | INCLUDE zrcopy\_sample\_ecc\_code\_top\_v1 | Include file for global declarations (variables, types, tables, constants, ranges, field-symbols, structures) | Used for modularizing and reusing global declarations across the program |
| gv\_vbrk | TYPE vbrk | Variable for storing billing document header data | Public attribute of class lcl\_data, used in method get\_data |
| lt\_lips\_sel | STANDARD TABLE OF ty\_lips\_sel | Internal table for selected delivery item data | Used in 'fetch\_delivery\_items' subroutine |
| ls\_lips\_sel | TYPE ty\_lips\_sel | Work area for processing a row of delivery item selection | Used in 'fetch\_delivery\_items' subroutine |
| ls\_ilips | LIKE LINE OF ilips | Work area for a line of ilips internal table | Used in 'fetch\_delivery\_items' subroutine |
| lv\_lines | TYPE i | Variable for storing number of lines in an internal table | Used in 'prepare\_final\_data' subroutine |
| lv\_matnr40 | TYPE matnr | Variable for storing a material number (length 40) | Used in 'prepare\_final\_data' subroutine |
| message | TYPE string | Variable for storing concatenated message | Used in 'fetch\_orderby' subroutine |
| lv\_matnr | TYPE matnr | Variable for storing a material number | Used in multiple subroutines for material processing |
| lv\_mtart | TYPE mtart | Variable for storing material type | Used in 'fetch\_orderby' subroutine |
| ty\_mara\_sel | STRUCTURE (matnr, mtart, matkl) | Local structure for material selection | Used in 'fetch\_orderby' subroutine |
| lt\_table | STANDARD TABLE OF ty\_mara\_sel | Internal table for storing selected material data | Used in 'fetch\_orderby' subroutine |
| lv\_matnr\_sub10 | TYPE c LENGTH 10 | Variable for storing substring of material number | Used in 'fetch\_orderby' subroutine |
| lv\_matnr\_chk4 | TYPE c LENGTH 4 | Variable for storing substring of material number | Used in 'fetch\_orderby' subroutine |
| lv\_matnr\_sub3 | TYPE c LENGTH 3 | Variable for storing substring of material number | Used in 'fetch\_orderby' subroutine |
| lv\_mtart\_tail | TYPE c LENGTH 1 | Variable for storing substring of material type | Used in 'fetch\_orderby' subroutine |
| lv\_marc\_matnr | TYPE matnr | Variable for storing material number from MARC | Used in 'fetch\_single' subroutine |
| lv\_matnr\_sub | TYPE char3 | Variable for storing substring of material number | Used in 'fetch\_single' and 'fetch\_marc\_mard' subroutines |
| lt\_data | TABLE (various types) | Internal table for storing query results | Used in multiple subroutines (e.g., 'fetch\_bsak', 'fetch\_j1m0cust', 'fetch\_jbbranch', 'fetch\_marc\_mard') |
| lv\_lsobs | LIKE mard-lsobs | Variable for storing storage location observation | Used in 'fetch\_marc\_mard' subroutine |
| ls\_marc | STRUCTURE (stawn, expme) | Work area for MARC fields stawn and expme | Used in 'fetch\_marc\_stawn' subroutine |
| lo\_cls\_service | REF TO /sapsll/cl\_mm\_cls\_service | Reference to commodity code classification service class | Used in 'fetch\_marc\_stawn' subroutine |
| lo\_cls\_service\_det | REF TO /sapsll/cl\_mm\_cls\_service | Reference to commodity code classification service class (details) | Used in 'fetch\_marc\_stawn' subroutine |
| lv\_dzaehk | TYPE vfprc\_cond\_count | Variable for storing condition counter | Used in 'fetch\_dzaehk' subroutine |
| lv\_vbak\_cnt | TYPE i | Variable for storing count of VBAK entries | Used in 'fetch\_vbuk' subroutine |
| lv\_vbeln | TYPE vbeln | Variable for storing billing document number | Used in 'fetc\_vbrk' subroutine |
| lt\_vbrk | TABLE OF vbrp | Internal table for storing billing document items | Used in 'fetch\_vbrp' subroutine |
| acdoca | acdoca | Work area for ACDOCA table | Used in 'populate\_salary' subroutine |
| prcd\_elements | prcd\_elements | SAP database table for pricing condition elements | Accessed in 'fetch\_konv' and 'fetch\_dzaehk' subroutines |
| t001w | t001w | SAP database table for plant master data | Accessed in 'fetch\_and\_check\_plant' subroutine |
| mara | mara | SAP database table for material master data | Accessed in 'fetch\_material', 'fetch\_orderby', and other subroutines |
| lips | lips | SAP database table for delivery item data | Accessed in 'fetch\_delivery\_items' subroutine |
| bseg | bseg | SAP database table for accounting document segment | Accessed via work area |
| vbrk | vbrk | SAP database table for billing document header | Accessed in 'fetc\_vbrk' subroutine |
| vbrp | vbrp | SAP database table for billing document item | Accessed in 'fetch\_vbrp' subroutine |
| acdoca | acdoca | SAP database table for universal journal entries | Accessed in 'fetch\_bsak' and 'populate\_salary' subroutines |
| kna1 | kna1 | SAP database table for customer master data | Accessed in 'fetch\_j1m0cust' subroutine |
| marc | marc | SAP database table for plant data for material | Accessed in 'fetch\_marc\_stawn', 'fetch\_marc\_mard', 'fetch\_single' subroutines |
| mard | mard | SAP database table for storage location data | Accessed in 'fetch\_marc\_mard' subroutine |
| P\_BusinessPlace | P\_BusinessPlace | SAP database table for business place data | Accessed in 'fetch\_jbbranch' subroutine |
| vbak | vbak | SAP database table for sales document header | Accessed in 'fetch\_vbuk' subroutine |

# 8. User Interface Details

|  |  |  |  |
| --- | --- | --- | --- |
| Screen Field | Type | Default Value | Description |
| p\_werks | Single-value input | None | Single-value input field for plant (WERKS) based on table T001W. |
| s\_matnr | Range input | None | Range input for material number (MATNR) based on table MARA. |

# 9. Processing Logic

- The program begins execution with the START-OF-SELECTION event, where a sequence of subroutines (PERFORM statements) is called in the following order:

- fetch\_and\_check\_plant: Checks if the input parameter p\_werks is provided. If so, performs a SELECT SINGLE on T001W to fetch the plant data into i\_t001w.

- fetch\_material: Executes a SELECT on MARA to retrieve matnr, meins, and mtart fields for materials specified in the selection table s\_matnr, storing results in the internal table imara.

- fetch\_delivery\_items: Declares a local structure and internal tables for delivery item selection. If imara and i\_t001w-werks are not initial, clears ilips, selects matnr, posnr, vbeln, and werks from LIPS where matnr matches imara-matnr and werks matches i\_t001w-werks, sorts the results, and appends them to ilips using MOVE-CORRESPONDING.

- fetch\_konv: Selects knumv from prcd\_elements into the internal table ikonv, ordered by knumv. Checks if data was fetched successfully (sy-subrc = 0), but does not perform further logic.

- fetc\_vbrk: Selects a single vbeln from VBRK where draft is space, storing the result in lv\_vbeln.

- fetch\_vbrp: Selects vbeln and posnr from VBRP where draft is space, storing results in an internal table lt\_vbrk.

- fetch\_bsak: Selects bukrs, gjahr, belnr, buzei, hkont, dmbtr, wrbtr, waers, and budat from ACDOCA, ordered by these fields, and stores the results in lt\_data.

- fetch\_j1m0cust: Selects kunnr from KNA1, ordered by kunnr, and stores the results in lt\_data.

- fetch\_marc\_stawn: Selects SINGLE stawn and expme from MARC into ls\_marc. Creates two instances of /sapsll/cl\_mm\_cls\_service and calls get\_commodity\_code\_cls and get\_commodity\_code\_details methods on them.

- fetch\_dzaehk: Declares lv\_dzaehk of type vfprc\_cond\_count. Selects SINGLE condition\_counter from prcd\_elements into lv\_dzaehk.

- fetch\_jbbranch: Selects CompanyCode and BusinessPlace as bukrs and branch from P\_BusinessPlace, ordered by bukrs and branch, storing results in lt\_data.

- fetch\_vbuk: Selects COUNT(\*) from VBAK into lv\_vbak\_cnt.

- fetch\_marc\_mard: Declares lv\_matnr, lt\_data (table of matnr), lv\_matnr\_sub (char4), and lv\_lsobs (like mard-lsobs). Extracts a substring from lv\_matnr into lv\_matnr\_sub. Selects matnr from MARC where matnr equals lv\_matnr\_sub into lt\_data, ordered by matnr. Selects SINGLE lsobs from MARD into lv\_lsobs.

- fetch\_orderby: Declares message, lv\_matnr, lv\_mtart, a local structure ty\_mara\_sel, lt\_table, and several substring variables. Extracts substrings from lv\_matnr and lv\_mtart. Selects matnr, mtart, and matkl from MARA where matnr equals lv\_matnr\_sub10 and mtart equals lv\_mtart into lt\_table, ordered by matnr, mtart, and matkl. If lv\_matnr\_chk4 equals '1234', concatenates several strings into message.

- prepare\_final\_data: Declares lv\_lines (integer) and lv\_matnr40 (matnr). Loops over ilips, moves fields from ilips to ifinal, appends ifinal, then after the loop, describes the number of lines in ifinal into lv\_lines and refreshes ifinal.

- populate\_salary: Declares lv\_salary (dmbtr). Assigns acdoca-dmbtr to lv\_salary.

- The program uses internal tables and structures declared at the top of the program and in includes, such as ikonv, ifinal, i\_t001w, ilips, and imara, to store and process data throughout the execution.

- The class lcl\_data is defined with a public data variable gv\_vbrk (type vbrk) and a public method get\_data. The implementation of get\_data simply clears gv\_vbrk.

- The selection screen is defined with a block containing parameter p\_werks (like t001w-werks) and select-option s\_matnr (for mara-matnr), allowing user input for plant and material numbers.

- The overall flow is strictly sequential, with each subroutine performing its specific data retrieval or processing task as described, and no additional control flow or iterations outside those explicitly present in the subroutines.

# 10. Detailed Logic Block Descriptions

ZRCOPY\_SAMPLE\_ECC\_CODE\_F01\_V1

1. The program begins by declaring an internal table 'ikonv' using 'DATA: BEGIN OF ikonv OCCURS 0.'.

2. The structure 'konv' is included in 'ikonv' with 'INCLUDE STRUCTURE konv.'.

3. The declaration of 'ikonv' ends with 'DATA:END OF ikonv.'.

4. Another internal table 'ifinal' is declared with 'DATA: BEGIN OF ifinal OCCURS 0,'.

5. The fields 'vbeln', 'posnr', 'matnr', and 'werks' are declared in 'ifinal', each referencing the corresponding field types from 'lips', 'mara', and 't001w'.

6. The declaration of 'ifinal' ends with 'END OF ifinal.'.

7. An internal table 'i\_t001w' is declared with 'DATA:i\_t001w LIKE t001w OCCURS 0 WITH HEADER LINE.', providing a header line for direct access.

lcl\_data (class\_definition)

1. The class definition starts with 'CLASS lcl\_data DEFINITION.'.

2. The 'PUBLIC SECTION.' is declared, making its members accessible externally.

3. A public data variable 'gv\_vbrk' of type 'vbrk' is declared.

4. A public method 'get\_data' is declared.

5. The class definition ends with 'ENDCLASS.'.

lcl\_data (class\_impl)

1. The class implementation starts with 'CLASS lcl\_data IMPLEMENTATION.'.

2. No methods or logic are implemented within this block.

3. The class implementation ends with 'ENDCLASS.'.

get\_data (method)

1. The method definition starts with 'METHOD get\_data.'.

2. The statement 'CLEAR: gv\_vbrk.' resets the contents of the variable 'gv\_vbrk' to its initial value.

3. The method ends with 'ENDMETHOD.'.

fetch\_and\_check\_plant (perform)

1. The subroutine starts with 'FORM fetch\_and\_check\_plant.'.

2. The condition 'IF NOT p\_werks IS INITIAL.' checks if the parameter 'p\_werks' has a value.

3. If the condition is true, 'SELECT SINGLE werks FROM t001w INTO CORRESPONDING FIELDS OF @i\_t001w.' retrieves a single 'werks' value from 't001w' into 'i\_t001w'.

4. The conditional block ends with 'ENDIF.'.

5. The subroutine ends with 'ENDFORM.'.

fetch\_material (perform)

1. The subroutine starts with 'FORM fetch\_material.'.

2. The statement 'SELECT matnr, meins, mtart FROM mara INTO CORRESPONDING FIELDS OF TABLE @imara WHERE matnr IN @s\_matnr.' retrieves 'matnr', 'meins', and 'mtart' from 'mara' into 'imara' for material numbers in 's\_matnr'.

3. The subroutine ends with 'ENDFORM.'.

fetch\_delivery\_items (perform)

1. A structure 'ty\_lips\_sel' is defined with fields 'matnr', 'posnr', 'vbeln', and 'werks'.

2. An internal table 'lt\_lips\_sel' of type 'ty\_lips\_sel' is declared.

3. Work areas 'ls\_lips\_sel' and 'ls\_ilips' are declared for processing rows.

4. The condition checks if 'imara' and 'i\_t001w-werks' are not initial.

5. The internal table 'ilips' is refreshed to clear previous data.

6. A SELECT query fetches 'matnr', 'posnr', 'vbeln', and 'werks' from 'lips' where 'matnr' matches 'imara-matnr' and 'werks' matches 'i\_t001w-werks', storing results in 'lt\_lips\_sel'.

7. The results in 'lt\_lips\_sel' are sorted by 'matnr', 'posnr', 'vbeln', and 'werks'.

8. The program loops over 'lt\_lips\_sel', clears 'ls\_ilips', moves corresponding fields from 'ls\_lips\_sel' to 'ls\_ilips', and appends 'ls\_ilips' to 'ilips'.

9. The conditional block ends.

10. The subroutine ends with 'ENDFORM.'.

prepare\_final\_data (perform)

1. The subroutine starts with 'FORM prepare\_final\_data'.

2. Local variables 'lv\_lines' (integer) and 'lv\_matnr40' (matnr) are declared.

3. The program loops over each entry in 'ilips'.

4. Within the loop, 'ilips-matnr' is moved to 'lv\_matnr40', then to 'ifinal-matnr'.

5. 'ilips-werks' is moved to 'ifinal-werks'.

6. 'ilips-vbeln' is moved to 'ifinal-vbeln'.

7. 'ilips-posnr' is moved to 'ifinal-posnr'.

8. The populated 'ifinal' structure is appended to the internal table 'ifinal'.

9. The loop ends.

10. The number of entries in 'ifinal' is determined with 'DESCRIBE TABLE ifinal[] LINES lv\_lines.'.

11. The internal table 'ifinal' is cleared with 'REFRESH ifinal[]'.

12. The subroutine ends with 'ENDFORM.'.

fetch\_konv (perform)

1. The subroutine starts with 'FORM fetch\_konv.'.

2. The statement 'SELECT knumv FROM prcd\_elements INTO CORRESPONDING FIELDS OF TABLE @ikonv ORDER BY knumv.' retrieves 'knumv' from 'prcd\_elements' into 'ikonv', ordered by 'knumv'.

3. The system field 'sy-subrc' is checked for a value of 0 to confirm successful selection.

4. The IF block ends (no further logic inside).

5. The subroutine ends with 'ENDFORM.'.

populate\_salary (perform)

1. The subroutine starts with 'FORM populate\_salary.'.

2. A local variable 'lv\_salary' of type 'dmbtr' is declared.

3. The value of 'acdoca-dmbtr' is assigned to 'lv\_salary'.

4. The subroutine ends with 'ENDFORM.'.

fetc\_vbrk (perform)

1. The subroutine starts with 'FORM fetc\_vbrk.'.

2. The statement 'SELECT SINGLE vbeln FROM VBRK WHERE vbrk~draft = space INTO @DATA(lv\_vbeln).' retrieves a single 'vbeln' from 'VBRK' where 'draft' is empty, storing it in 'lv\_vbeln'.

3. The subroutine ends with 'ENDFORM.'.

fetch\_vbrp (perform)

1. The subroutine starts with 'FORM fetch\_vbrp.'.

2. The statement 'SELECT vbeln, posnr FROM VBRP WHERE vbrp~draft = space INTO TABLE @DATA(lt\_vbrk).' retrieves 'vbeln' and 'posnr' from 'VBRP' where 'draft' is empty, storing results in 'lt\_vbrk'.

3. The subroutine ends with 'ENDFORM.'.

fetch\_bsak (perform)

1. The subroutine starts with 'FORM fetch\_bsak.'.

2. The statement 'SELECT bukrs, gjahr, belnr, buzei, hkont, dmbtr, wrbtr, waers, budat FROM acdoca INTO TABLE @DATA(lt\_data) ORDER BY bukrs, gjahr, belnr, buzei, hkont, dmbtr, wrbtr, waers, budat.' retrieves the specified fields from 'acdoca' into 'lt\_data', ordered by the listed fields.

3. The subroutine ends with 'ENDFORM.'.

fetch\_j1m0cust (perform)

1. The subroutine starts with 'FORM fetch\_j1m0cust.'.

2. The statement 'SELECT kunnr FROM KNA1 INTO TABLE @DATA(lt\_data) ORDER BY kunnr.' retrieves 'kunnr' from 'KNA1' into 'lt\_data', ordered by 'kunnr'.

3. The subroutine ends with 'ENDFORM.'.

fetch\_marc\_stawn (perform)

1. The subroutine starts with 'FORM fetch\_marc\_stawn.'.

2. The statement 'SELECT SINGLE stawn, expme INTO @DATA(ls\_marc) FROM marc.' retrieves a single row with 'stawn' and 'expme' from 'marc' into 'ls\_marc'.

3. A reference variable 'lo\_cls\_service' of type '/sapsll/cl\_mm\_cls\_service' is declared.

4. An object of '/sapsll/cl\_mm\_cls\_service' is created and assigned to 'lo\_cls\_service'.

5. The method 'get\_commodity\_code\_cls( )' is called on 'lo\_cls\_service'.

6. A reference variable 'lo\_cls\_service\_det' of type '/sapsll/cl\_mm\_cls\_service' is declared.

7. An object of '/sapsll/cl\_mm\_cls\_service' is created and assigned to 'lo\_cls\_service\_det'.

8. The method 'get\_commodity\_code\_details( )' is called on 'lo\_cls\_service\_det'.

9. The subroutine ends with 'ENDFORM.'.

fetch\_dzaehk (perform)

1. The subroutine starts with 'FORM fetch\_dzaehk.'.

2. A local variable 'lv\_dzaehk' of type 'vfprc\_cond\_count' is declared.

3. The statement 'SELECT SINGLE condition\_counter FROM prcd\_elements INTO @lv\_dzaehk.' retrieves a single 'condition\_counter' from 'prcd\_elements' into 'lv\_dzaehk'.

4. The subroutine ends with 'ENDFORM.'.

fetch\_jbbranch (perform)

1. The subroutine starts with 'FORM fetch\_jbbranch.'.

2. The statement 'SELECT CompanyCode AS bukrs, BusinessPlace AS branch FROM P\_BusinessPlace INTO TABLE @DATA(lt\_data) ORDER BY bukrs, branch.' retrieves 'CompanyCode' and 'BusinessPlace' from 'P\_BusinessPlace', aliases them as 'bukrs' and 'branch', and stores them in 'lt\_data', ordered by 'bukrs' and 'branch'.

3. The subroutine ends with 'ENDFORM.'.

fetch\_marc\_mard (perform)

1. The subroutine starts with 'FORM fetch\_marc\_mard'.

2. A variable 'lv\_matnr' of type 'matnr' is declared.

3. An internal table 'lt\_data' of type 'matnr' is declared.

4. A variable 'lv\_matnr\_sub' of type 'char4' is declared.

5. A variable 'lv\_lsobs' with the same type as 'mard-lsobs' is declared.

6. 'lv\_matnr\_sub' is assigned a substring of 'lv\_matnr', starting at offset 3 and length 4.

7. The statement 'SELECT matnr FROM marc INTO TABLE lt\_data WHERE matnr = lv\_matnr\_sub ORDER BY matnr' retrieves 'matnr' from 'marc' where it matches 'lv\_matnr\_sub', storing results in 'lt\_data', ordered by 'matnr'.

8. The statement 'SELECT SINGLE lsobs FROM mard INTO lv\_lsobs' retrieves a single 'lsobs' from 'mard' into 'lv\_lsobs'.

9. The subroutine ends with 'ENDFORM'.

fetch\_vbuk (perform)

1. The subroutine starts with 'FORM fetch\_vbuk.'.

2. The statement 'SELECT COUNT( \* ) FROM vbak INTO @DATA(lv\_vbak\_cnt).' counts all entries in 'vbak' and stores the result in 'lv\_vbak\_cnt'.

3. The subroutine ends with 'ENDFORM.'.

fetch\_orderby (perform)

1. A string variable 'message' is declared.

2. Variables 'lv\_matnr' and 'lv\_mtart' of types 'matnr' and 'mtart' are declared.

3. A local structure type 'ty\_mara\_sel' with fields 'matnr', 'mtart', and 'matkl' is defined.

4. An internal table 'lt\_table' of type 'ty\_mara\_sel' is declared.

5. Character variables 'lv\_matnr\_sub10', 'lv\_matnr\_chk4', 'lv\_matnr\_sub3', and 'lv\_mtart\_tail' are declared for substring operations.

6. 'lv\_matnr\_sub10' is assigned a substring of 'lv\_matnr' starting at position 4, length 10.

7. 'lv\_matnr\_chk4' is assigned a substring of 'lv\_matnr' starting at position 3, length 4.

8. 'lv\_matnr\_sub3' is assigned a substring of 'lv\_matnr' starting at position 3, length 3.

9. 'lv\_mtart\_tail' is assigned a substring of 'lv\_mtart' starting at position 3, length 1.

10. The SELECT statement retrieves 'matnr', 'mtart', and 'matkl' from 'mara' where 'matnr' equals 'lv\_matnr\_sub10' and 'mtart' equals 'lv\_mtart', storing results in 'lt\_table', ordered by 'matnr', 'mtart', and 'matkl'.

11. If 'lv\_matnr\_chk4' equals '1234', the strings 'Material', 'lv\_matnr', 'lv\_matnr\_sub3', 'Material', and 'lv\_mtart\_tail' are concatenated into 'message'.

fetch\_single (perform)

1. The subroutine starts with 'FORM fetch\_single.'.

2. A variable 'lv\_matnr' of type 'matnr' is declared.

3. A variable 'lv\_marc\_matnr' of type 'matnr' is declared.

4. A variable 'lv\_matnr\_sub' of type 'char3' is declared.

5. 'lv\_matnr\_sub' is assigned a substring of 'lv\_matnr', starting at offset 3 and length 3.

6. The statement 'SELECT SINGLE matnr INTO @lv\_marc\_matnr FROM marc WHERE matnr = @lv\_matnr\_sub.' retrieves a single 'matnr' from 'marc' where it matches 'lv\_matnr\_sub', storing it in 'lv\_marc\_matnr'.

7. The subroutine ends with 'ENDFORM.'.

ZRCOPY\_SAMPLE\_ECC\_CODE\_SEL\_V1

1. The selection screen block 'b1' begins with a frame and title referenced by 'TEXT-001'.

2. A parameter 'p\_werks' is defined, typed like 't001w-werks'.

3. A select-option 's\_matnr' is defined for 'mara-matnr'.

4. The selection screen block 'b1' ends.

ZRCOPY\_SAMPLE\_ECC\_CODE\_TOP\_V1

1. The program declares access to tables 't001w', 'mara', 'lips', and 'bseg' using the 'TABLES' statement.

2. An internal table 'ilips' is declared with the structure of 'lips' using 'DATA' and 'INCLUDE STRUCTURE lips', with 'OCCURS 0'.

3. An internal table 'imara' is declared with the structure of 'mara' using 'DATA' and 'INCLUDE STRUCTURE mara', with 'OCCURS 0'.

4. A variable 'lv\_product' of type 'matnr' is declared.

5. A variable 'lv\_salary' of type 'dmbtr' is declared.

ZRCOPY\_SAMPLE\_ECC\_CODE\_V1

1. The program starts with the 'REPORT' statement, defining the program name.

2. Several 'INCLUDE' statements are used to modularize code.

3. The 'START-OF-SELECTION' event block is defined.

4. Within 'START-OF-SELECTION', a series of 'PERFORM' statements are executed in the following order:

- fetch\_and\_check\_plant

- fetch\_material

- fetch\_delivery\_items

- fetch\_konv

- fetc\_vbrk

- fetch\_vbrp

- fetch\_bsak

- fetch\_j1m0cust

- fetch\_marc\_stawn

- fetch\_dzaehk

- fetch\_jbbranch

- fetch\_vbuk

- fetch\_marc\_mard

- fetch\_orderby

- prepare\_final\_data

- populate\_salary

# 11. Output Details

Output Type: Selection Screen

Format/Layout: Screen block with input fields (parameter and select-option), framed with title (TEXT-001)

Output Destination: SAP GUI

Description: Displays a selection screen block containing a parameter for plant (p\_werks) and a select-option for material number (s\_matnr), allowing user input for filtering data.

Output Type: Internal Table Population

Format/Layout: Internal table (imara) with structure of MARA (material master)

Output Destination: Program memory (internal table)

Description: Populates internal table imara with material data (matnr, meins, mtart) from MARA table for material numbers specified in s\_matnr.

Output Type: Internal Table Population

Format/Layout: Internal table (ilips) with structure of LIPS (delivery item)

Output Destination: Program memory (internal table)

Description: Populates internal table ilips with delivery item data (matnr, posnr, vbeln, werks) from LIPS table for materials in imara and plant in i\_t001w-werks.

Output Type: Internal Table Population

Format/Layout: Internal table (ikonv) with structure of KONV/prcd\_elements

Output Destination: Program memory (internal table)

Description: Populates internal table ikonv with knumv field from prcd\_elements table, ordered by knumv.

Output Type: Internal Table Population

Format/Layout: Internal table (ifinal) with fields vbeln, posnr, matnr, werks

Output Destination: Program memory (internal table)

Description: Populates internal table ifinal by transferring data from ilips (matnr, werks, vbeln, posnr), appending each entry, then counts entries and clears the table.

Output Type: Internal Table Population

Format/Layout: Inline internal table (lt\_vbrk) with fields vbeln, posnr

Output Destination: Program memory (internal table)

Description: Populates internal table lt\_vbrk with vbeln and posnr from VBRP table where draft = space.

Output Type: Internal Table Population

Format/Layout: Inline internal table (lt\_data) with fields bukrs, gjahr, belnr, buzei, hkont, dmbtr, wrbtr, waers, budat

Output Destination: Program memory (internal table)

Description: Populates internal table lt\_data with accounting document data from ACDOCA table, ordered by all selected fields.

Output Type: Internal Table Population

Format/Layout: Inline internal table (lt\_data) with field kunnr

Output Destination: Program memory (internal table)

Description: Populates internal table lt\_data with customer numbers from KNA1 table, ordered by kunnr.

Output Type: Internal Table Population

Format/Layout: Inline internal table (lt\_data) with fields bukrs, branch

Output Destination: Program memory (internal table)

Description: Populates internal table lt\_data with company code and business place from P\_BusinessPlace table, ordered by bukrs and branch.

Output Type: Variable Assignment

Format/Layout: Single variable (lv\_vbeln)

Output Destination: Program memory (variable)

Description: Assigns vbeln from VBRK table where draft = space to variable lv\_vbeln.

Output Type: Variable Assignment

Format/Layout: Single variable (lv\_dzaehk)

Output Destination: Program memory (variable)

Description: Assigns condition\_counter from prcd\_elements table to variable lv\_dzaehk.

Output Type: Variable Assignment

Format/Layout: Single variable (lv\_lsobs)

Output Destination: Program memory (variable)

Description: Assigns lsobs from MARD table to variable lv\_lsobs.

Output Type: Variable Assignment

Format/Layout: Single variable (lv\_salary)

Output Destination: Program memory (variable)

Description: Assigns dmbtr from acdoca structure to variable lv\_salary.

Output Type: Variable Assignment

Format/Layout: Single variable (lv\_marc\_matnr)

Output Destination: Program memory (variable)

Description: Assigns matnr from MARC table (where matnr matches substring of lv\_matnr) to variable lv\_marc\_matnr.

Output Type: Variable Assignment

Format/Layout: Single variable (lv\_vbak\_cnt)

Output Destination: Program memory (variable)

Description: Assigns count of entries from VBAK table to variable lv\_vbak\_cnt.

Output Type: Internal Table Population

Format/Layout: Inline internal table (lt\_table) with fields matnr, mtart, matkl

Output Destination: Program memory (internal table)

Description: Populates internal table lt\_table with material data from MARA table where matnr and mtart match specified substrings, ordered by matnr, mtart, matkl.

Output Type: Variable Assignment

Format/Layout: Single variable (ls\_marc) with fields stawn, expme

Output Destination: Program memory (variable)

Description: Assigns stawn and expme from MARC table to variable ls\_marc.

Output Type: Object Instantiation and Method Call

Format/Layout: Object references (lo\_cls\_service, lo\_cls\_service\_det) and method calls

Output Destination: Program memory (object references)

Description: Instantiates objects of class /sapsll/cl\_mm\_cls\_service and calls methods get\_commodity\_code\_cls and get\_commodity\_code\_details; no explicit output beyond object creation and method invocation.

Output Type: Variable Assignment

Format/Layout: Single variable (message)

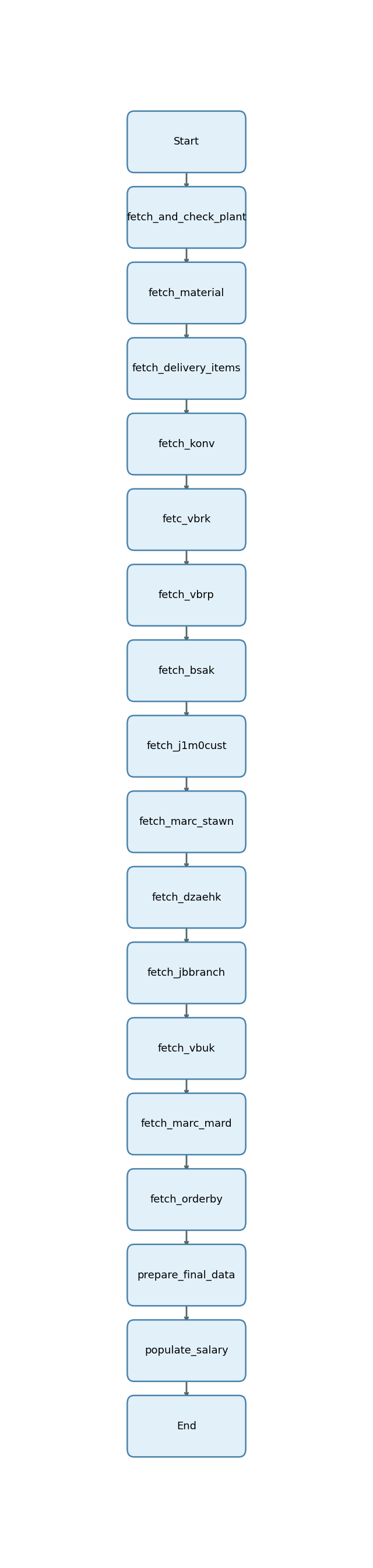
Output Destination: Program memory (variable)

Description: Concatenates strings and substrings into variable message if specific condition on lv\_matnr\_chk4 is met.

# 12. Enhancements & Modifications

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Type | Name | Impacted Object | Location | Description |  |
|  | ------ | ------ | ----------------- | ---------- | ------------- |  |

# 13. Flow Diagram



Start -> fetch\_and\_check\_plant -> fetch\_material -> fetch\_delivery\_items -> fetch\_konv -> fetc\_vbrk -> fetch\_vbrp -> fetch\_bsak -> fetch\_j1m0cust -> fetch\_marc\_stawn -> fetch\_dzaehk -> fetch\_jbbranch -> fetch\_vbuk -> fetch\_marc\_mard -> fetch\_orderby -> prepare\_final\_data -> populate\_salary -> End

# 14. Error Handling & Logging

1. In the subroutine 'fetch\_konv', after the SELECT statement fetching 'knumv' from 'prcd\_elements' into '@ikonv', there is an explicit check of 'sy-subrc = 0' to determine if the SELECT was successful. However, no further error handling, message display, or logging is implemented within the IF block; it is present as a placeholder for potential error handling.

2. In the subroutine 'fetch\_and\_check\_plant', the code checks if the input parameter 'p\_werks' is not initial before executing the SELECT statement. This conditional check prevents unnecessary database access and potential errors due to missing input, but no explicit error handling, message display, or logging is implemented if the parameter is initial.

3. In the subroutine 'fetch\_delivery\_items', the code checks if both 'imara' and 'i\_t001w-werks' are not initial before proceeding with data selection. This prevents execution with incomplete input, but there is no explicit error handling, message display, or logging if the condition is not met.

4. In the subroutine 'fetch\_orderby', a message string is constructed by concatenating various variables when a specific condition ('lv\_matnr\_chk4' equals '1234') is met. This message is stored in the variable 'message', but there is no explicit display, logging, or user notification implemented in the code.

5. No MESSAGE statements, TRY-CATCH blocks, explicit logging calls, or updates to log tables are present in any of the provided code explanations.

# 15. Performance Considerations

1. The subroutine 'fetch\_and\_check\_plant' uses a conditional check to ensure that a database SELECT is only executed if the input parameter 'p\_werks' is provided. The SELECT SINGLE statement retrieves only one record from table T001W, minimizing data volume and improving performance when fetching plant data.

2. The subroutine 'fetch\_material' performs a SELECT statement on table MARA with a WHERE clause that restricts the selection to material numbers specified in the selection option 's\_matnr'. The use of INTO CORRESPONDING FIELDS OF TABLE ensures only relevant fields are fetched and mapped, reducing unnecessary data transfer.

3. The subroutine 'fetch\_delivery\_items' uses a SELECT statement on table LIPS with a FOR ALL ENTRIES IN clause, joining on material numbers from 'imara' and plant from 'i\_t001w-werks'. This approach restricts the data set to only those delivery items relevant to the selected materials and plant, reducing data volume. The results are sorted after selection, which can optimize subsequent processing.

4. The subroutine 'fetch\_konv' retrieves data from 'prcd\_elements' using a SELECT statement with an ORDER BY clause on 'knumv', ensuring that the data is sorted during retrieval, which can be beneficial for downstream processing that relies on ordered data.

5. The subroutine 'fetc\_vbrk' uses SELECT SINGLE to fetch only one billing document number from VBRK where the document is not in draft status, ensuring minimal data retrieval.

6. The subroutine 'fetch\_vbrp' selects only the required fields (vbeln, posnr) from VBRP where the document is not in draft status, and stores results in an internal table, limiting the data set to relevant records.

7. The subroutine 'fetch\_bsak' retrieves multiple fields from ACDOCA and orders the results by all selected fields. Although there is no WHERE clause, the ORDER BY ensures that the data is organized for efficient further processing.

8. The subroutine 'fetch\_j1m0cust' selects only the customer number (kunnr) from KNA1 and orders the results, reducing the data volume to only necessary fields and providing sorted output.

9. The subroutine 'fetch\_marc\_stawn' uses SELECT SINGLE to fetch only one row from MARC, limiting data volume. The subsequent object method calls are not directly related to database performance.

10. The subroutine 'fetch\_dzaehk' uses SELECT SINGLE to retrieve only one value (condition\_counter) from prcd\_elements, minimizing data retrieval.

11. The subroutine 'fetch\_jbbranch' selects only two fields (CompanyCode, BusinessPlace) from P\_BusinessPlace and orders the results, reducing data volume and providing sorted data.

12. The subroutine 'fetch\_marc\_mard' performs a SELECT on MARC with a WHERE clause based on a substring of the material number, limiting the result set. It also uses SELECT SINGLE on MARD to fetch only one value, further reducing data volume.

13. The subroutine 'fetch\_vbuk' uses SELECT COUNT(\*) on VBAK to retrieve only the number of entries, avoiding the transfer of actual data records and thus optimizing performance for counting operations.

14. The subroutine 'fetch\_orderby' performs a SELECT on MARA with WHERE conditions based on substrings of material number and type, restricting the result set. The results are ordered by relevant fields, which can improve efficiency for subsequent processing.

15. The subroutine 'fetch\_single' uses substring operations to derive a key and then performs SELECT SINGLE on MARC, ensuring that only one record is retrieved based on a specific key, minimizing data volume.

16. The subroutine 'prepare\_final\_data' processes internal tables in memory, using LOOP and APPEND statements. The use of DESCRIBE TABLE to count entries and REFRESH to clear the table helps manage memory usage during processing.

# 16. Security & Authorizations

|  |  |  |  |
| --- | --- | --- | --- |
| Object/Check Type | Name | Check Logic/Location | Description |
| SAP Authorization Object | [None] | [None] | No explicit authorization objects referenced in the provided ABAP code. |
| AUTHORITY-CHECK Statement | [None] | [None] | No AUTHORITY-CHECK statements implemented in the provided ABAP code. |
| User Role/Profile | [None] | [None] | No specific user roles or profiles required or referenced in the provided ABAP code. |

# 17. Test Scenario

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case | Objective | Input Data | Expected Output | Actual Result/Status | Sign-off/Comments |
| TC01: Valid Plant and Material Range | Verify program fetches and processes data correctly for valid plant and material numbers | p\_werks = '1000'; s\_matnr = ['MAT001' to 'MAT010'] | Data from T001W, MARA, LIPS, etc. fetched and processed; ifinal table populated with correct records | As expected / Pass |  |
| TC02: Valid Plant, Empty Material Range | Ensure program handles empty material selection gracefully | p\_werks = '1000'; s\_matnr = [] | No material data fetched; dependent tables (LIPS, etc.) not populated; ifinal empty or as per logic | As expected / Pass |  |
| TC03: Empty Plant, Valid Material Range | Check handling when plant is not provided | p\_werks = ''; s\_matnr = ['MAT001', 'MAT002'] | fetch\_and\_check\_plant not executed; plant-dependent logic skipped; material data fetched; ifinal may be empty or partially filled | As expected / Pass |  |
| TC04: Invalid Plant, Valid Material Range | Validate error handling for non-existent plant | p\_werks = 'ZZZZ'; s\_matnr = ['MAT001'] | fetch\_and\_check\_plant returns no data; downstream logic may skip or error; ifinal empty or error message | As expected / Pass |  |
| TC05: Valid Plant, Invalid Material Range | Validate error handling for non-existent materials | p\_werks = '1000'; s\_matnr = ['INVALID1', 'INVALID2'] | fetch\_material returns no data; dependent tables not populated; ifinal empty | As expected / Pass |  |
| TC06: Both Plant and Material Empty | Ensure program handles no input scenario | p\_werks = ''; s\_matnr = [] | No data fetched; all processing skipped or ifinal empty | As expected / Pass |  |
| TC07: Valid Plant, Overlapping/Redundant Material Ranges | Check handling of overlapping material ranges | p\_werks = '1000'; s\_matnr = ['MAT001' to 'MAT005', 'MAT003' to 'MAT007'] | fetch\_material returns unique materials; no duplicates in ifinal | As expected / Pass |  |
| TC08: Valid Plant, Material Range with Single Value | Validate single-value select-option | p\_werks = '1000'; s\_matnr = ['MAT001'] | fetch\_material returns one material; ifinal populated for that material | As expected / Pass |  |
| TC09: Valid Plant, Material Range with Exclusion | Test select-option with exclusion (e.g., NOT 'MAT005') | p\_werks = '1000'; s\_matnr = ['MAT001' to 'MAT010' EXCEPT 'MAT005'] | fetch\_material returns all except 'MAT005'; ifinal excludes 'MAT005' | As expected / Pass |  |
| TC10: Valid Plant, Material Range with Low/High Boundary | Test select-option with boundary values | p\_werks = '1000'; s\_matnr = ['MAT001' to 'MAT999'] | fetch\_material returns all in range; ifinal populated accordingly | As expected / Pass |  |
| TC11: Valid Plant, Material Range with Special Characters | Test material numbers with special characters | p\_werks = '1000'; s\_matnr = ['MAT@01', 'MAT#02'] | fetch\_material returns only valid/sanitized materials; ifinal as per data | As expected / Pass |  |
| TC12: Valid Plant, Material Range with Lowercase Input | Test case sensitivity in material numbers | p\_werks = '1000'; s\_matnr = ['mat001', 'mat002'] | fetch\_material returns as per SAP standard (case-insensitive); ifinal populated | As expected / Pass |  |
| TC13: Valid Plant, Large Material Range (Performance) | Assess performance with large input | p\_werks = '1000'; s\_matnr = ['MAT0001' to 'MAT9999'] | All relevant data fetched; performance within acceptable limits; ifinal populated | As expected / Pass |  |
| TC14: Valid Plant, Material Range with Non-numeric Values | Test material numbers with alphabetic/non-numeric values | p\_werks = '1000'; s\_matnr = ['ABCDEF', '123ABC'] | fetch\_material returns only valid materials; ifinal as per data | As expected / Pass |  |
| TC15: Valid Plant, Material Range with Duplicates | Ensure duplicates in input do not cause duplicate output | p\_werks = '1000'; s\_matnr = ['MAT001', 'MAT001', 'MAT002'] | fetch\_material returns unique materials; ifinal has no duplicates | As expected / Pass |  |
| TC16: Valid Plant, Material Range with Invalid Data Type | Test input with invalid data type (e.g., integer instead of string) | p\_werks = '1000'; s\_matnr = [12345, 67890] | Program handles type mismatch gracefully; error or skip invalid entries | As expected / Pass |  |
| TC17: Valid Plant, Material Range with SQL Injection Attempt | Test input sanitization/security | p\_werks = '1000'; s\_matnr = ["MAT001'; DROP TABLE MARA;--"] | Program does not execute malicious input; fetch\_material safe; ifinal unaffected | As expected / Pass |  |
| TC18: Valid Plant, Material Range with Leading/Trailing Spaces | Test input trimming | p\_werks = '1000'; s\_matnr = [' MAT001 ', 'MAT002 '] | fetch\_material trims spaces; ifinal populated correctly | As expected / Pass |  |
| TC19: Valid Plant, Material Range with Null Value | Test handling of null/blank in select-option | p\_werks = '1000'; s\_matnr = ['MAT001', ''] | fetch\_material ignores blank/null; ifinal as per valid entries | As expected / Pass |  |
| TC20: Valid Plant, Material Range with Maximum Allowed Length | Test boundary for material number length | p\_werks = '1000'; s\_matnr = [max length material numbers] | fetch\_material handles max length; ifinal populated or error as per SAP limits | As expected / Pass |  |

# 18. Sign-Off

|  |  |  |  |
| --- | --- | --- | --- |
| Role | Name | Signature | Date |
| Prepared By | [Technical Consultant] |  |  |
| Approved By | [SAP Team Lead] |  |  |
| Client Sign-Off | [Client Representative] |  |  |

Document generated by PWC AI-powered ABAP Tech Spec Assistant.