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 |
| Technical Specification: ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 | ECC (Version as per system) | [To be filled] | PWC AI Asset | 1.0 / [Date] |

# 2. Introduction

This document provides the technical specification for the SAP ABAP program ZRCOPY\_SAMPLE\_ECC\_CODE\_V1. The objective is to detail the program's purpose, scope, and intended audience. The document outlines the program's structure, key components, and logic, serving as a reference for ABAP developers, technical consultants, and project stakeholders involved in the development, review, or maintenance of this solution. The scope covers the main program, its includes, selection screens, subroutines, and object-oriented components, ensuring a comprehensive understanding of the technical implementation.

# 3. Transport Management

|  |  |  |  |
| --- | --- | --- | --- |
| Development Package | Transport Request Number | Sequence/Dependency | Description |
| [To Be Filled] | [To Be Filled] | [To Be Filled] | [To Be Filled] |

# 4. Requirement Overview

The business requirement addressed by this ABAP development is to automate the extraction, processing, and consolidation of key business data from various SAP modules, including materials management, sales and distribution, and finance. The solution aims to provide a structured and efficient mechanism for retrieving plant, material, delivery, billing, customer, and financial data, as well as performing necessary validations and data transformations. The objective is to streamline reporting, data analysis, and downstream processing by ensuring accurate and timely data retrieval and preparation, supporting business operations and decision-making.

# 5. Solution Approach

1. Modular Program Structure: The solution is architected using a main ABAP executable program (ZRCOPY\_SAMPLE\_ECC\_CODE\_V1) with multiple includes for separation of concerns, such as selection screen definitions, data declarations, and subroutine implementations. This modularity enhances maintainability and reusability.

2. Subroutine-Based Data Processing: Key business logic is encapsulated within FORM routines (PERFORMs), each responsible for a specific data retrieval or processing task (e.g., fetching plant, material, delivery, billing, and financial data). This approach promotes clarity, reusability, and ease of testing.

3. Object-Oriented Enhancements: The solution incorporates local class definitions (e.g., lcl\_data) and methods (e.g., get\_data) to encapsulate related data and operations, supporting extensibility and alignment with modern ABAP development practices.

4. Dynamic Data Handling: Internal tables and structures are used extensively to store and process data retrieved from standard SAP tables. Data is filtered, sorted, and transformed as needed to meet reporting and processing requirements.

5. User-Driven Selection: The selection screen enables users to specify input parameters (such as plant and material numbers), ensuring that data extraction is flexible and aligned with user needs.

6. Comprehensive Data Coverage: The program orchestrates the retrieval of data from a wide range of SAP tables (e.g., MARA, LIPS, VBRK, VBRP, ACDOCA, KNA1, MARC, MARD, PRCD\_ELEMENTS), ensuring all relevant business data is included in the final output.

# 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 extraction and processing | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| Include | ZRCOPY\_SAMPLE\_ECC\_CODE\_F01\_V1 | Contains subroutine (FORM) implementations | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| Include | ZRCOPY\_SAMPLE\_ECC\_CODE\_SEL\_V1 | Contains selection screen definitions | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| Include | ZRCOPY\_SAMPLE\_ECC\_CODE\_TOP\_V1 | Contains data declarations and internal tables | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| Class (Local) | lcl\_data | Local class for encapsulating data and methods | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| Method | get\_data | Method of lcl\_data class for clearing billing document variable | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| FORM Routine | fetch\_and\_check\_plant | Fetches and validates plant data from T001W | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| FORM Routine | fetch\_material | Fetches material data from MARA based on user selection | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| FORM Routine | fetch\_delivery\_items | Retrieves delivery item data from LIPS based on material and plant | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| FORM Routine | prepare\_final\_data | Processes and consolidates data into the final output internal table | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| FORM Routine | fetch\_konv | Fetches condition records from PRCD\_ELEMENTS | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| FORM Routine | populate\_salary | Populates salary value from ACDOCA | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| FORM Routine | fetc\_vbrk | Fetches billing document header (VBELN) from VBRK | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| FORM Routine | fetch\_vbrp | Fetches billing document items from VBRP | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| FORM Routine | fetch\_bsak | Fetches financial data from ACDOCA | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| FORM Routine | fetch\_j1m0cust | Fetches customer data from KNA1 | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| FORM Routine | fetch\_marc\_stawn | Fetches MARC data and calls commodity code classification services | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| FORM Routine | fetch\_dzaehk | Fetches condition counter from PRCD\_ELEMENTS | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| FORM Routine | fetch\_jbbranch | Fetches business place data from P\_BusinessPlace | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| FORM Routine | fetch\_marc\_mard | Fetches material and storage location data from MARC and MARD | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| FORM Routine | fetch\_vbuk | Counts entries in VBAK (sales document headers) | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| FORM Routine | fetch\_orderby | Fetches and processes material data from MARA with ordering and message construction | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| FORM Routine | fetch\_single | Fetches a single material number from MARC based on substring logic | 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 | Global internal table 'ikonv' declared with structure KONV and header line. | Used to store condition records, populated in subroutine 'fetch\_konv' from table 'prcd\_elements'. |
| ifinal | Internal Table (with header line) with fields vbeln, posnr, matnr, werks | Global internal table 'ifinal' with custom structure containing fields vbeln, posnr, matnr, werks and header line. | Used to store final processed data, populated in 'prepare\_final\_data' subroutine. |
| i\_t001w | Internal Table (with header line) of type T001W | Global internal table 'i\_t001w' of type T001W with header line. | Used to store plant data, populated in 'fetch\_and\_check\_plant' subroutine. |
| t001w | t001w | Table work area declaration for database table T001W. | Used as a work area for plant data operations. |
| mara | mara | Table work area declaration for database table MARA. | Used as a work area for material master data operations. |
| lips | lips | Table work area declaration for database table LIPS. | Used as a work area for delivery item data operations. |
| bseg | bseg | Table work area declaration for database table BSEG. | Used as a work area for accounting document segment data operations. |
| ilips | OCCURS 0 structure based on LIPS | Internal table with header line based on structure LIPS. | Used to store delivery item data, populated in 'fetch\_delivery\_items' subroutine. |
| imara | OCCURS 0 structure based on MARA | Internal table with header line based on structure MARA. | Used to store material master data, populated in 'fetch\_material' subroutine. |
| lv\_product | TYPE matnr | Variable of type MATNR for storing a product number. | Used for temporary storage of material number in various subroutines. |
| lv\_salary | TYPE dmbtr | Variable of type DMBTR for storing a salary amount. | Used for temporary storage of salary amount, e.g., in 'populate\_salary' subroutine. |
| zrcopy\_sample\_ecc\_code\_top\_v1 | INCLUDE zrcopy\_sample\_ecc\_code\_top\_v1 | Includes global declarations such as variables, types, tables, constants, ranges, field-symbols, and structures from the top include file. | Used to modularize and centralize global declarations for reuse across the program. |
| t001w (SAP Table) | T001W | Plant master data table. | Read in 'fetch\_and\_check\_plant' subroutine and as work area. |
| mara (SAP Table) | MARA | Material master data table. | Read in 'fetch\_material', 'fetch\_orderby', 'fetch\_single', and as work area. |
| lips (SAP Table) | LIPS | Delivery item data table. | Read in 'fetch\_delivery\_items' and as work area. |
| bseg (SAP Table) | BSEG | Accounting document segment data table. | Used as work area for accounting data operations. |
| prcd\_elements (SAP Table) | PRCD\_ELEMENTS | Pricing condition elements table. | Read in 'fetch\_konv' and 'fetch\_dzaehk' subroutines. |
| vbrk (SAP Table) | VBRK | Billing document header table. | Read in 'fetc\_vbrk' subroutine. |
| vbrp (SAP Table) | VBRP | Billing document item table. | Read in 'fetch\_vbrp' subroutine. |
| acdoca (SAP Table) | ACDOCA | Universal journal entry line items table. | Read in 'fetch\_bsak' and 'populate\_salary' subroutines. |
| kna1 (SAP Table) | KNA1 | General customer master data table. | Read in 'fetch\_j1m0cust' subroutine. |
| marc (SAP Table) | MARC | Plant data for material table. | Read in 'fetch\_marc\_stawn', 'fetch\_marc\_mard', and 'fetch\_single' subroutines. |
| mard (SAP Table) | MARD | Storage location data for material table. | Read in 'fetch\_marc\_mard' subroutine. |
| vbak (SAP Table) | VBAK | Sales document: Header data table. | Read in 'fetch\_vbuk' subroutine. |
| P\_BusinessPlace (SAP Table) | P\_BusinessPlace | Business place master data table. | Read in 'fetch\_jbbranch' subroutine. |
| /sapsll/cl\_mm\_cls\_service | ABAP Class Reference | Commodity code classification service class. | Instantiated and used in 'fetch\_marc\_stawn' subroutine for commodity code operations. |
| vfprc\_cond\_count | Data Element | Data type for condition counter. | Used as type for 'lv\_dzaehk' in 'fetch\_dzaehk' subroutine. |
| vbrk~draft, vbrp~draft | Field | Draft status indicator fields in VBRK and VBRP tables. | Used as selection criteria in 'fetc\_vbrk' and 'fetch\_vbrp' subroutines. |
| lsobs | Field in MARD | Storage location stock indicator. | Used in 'fetch\_marc\_mard' subroutine. |
| matnr, meins, mtart, matkl | Fields in MARA | Material number, base unit, material type, material group. | Used in 'fetch\_material', 'fetch\_orderby', and related subroutines. |
| knumv, condition\_counter | Fields in PRCD\_ELEMENTS | Condition record number, condition counter. | Used in 'fetch\_konv' and 'fetch\_dzaehk' subroutines. |
| bukrs, gjahr, belnr, buzei, hkont, dmbtr, wrbtr, waers, budat | Fields in ACDOCA | Company code, fiscal year, document number, line item, G/L account, amounts, currency, posting date. | Used in 'fetch\_bsak' subroutine. |
| kunnr | Field in KNA1 | Customer number. | Used in 'fetch\_j1m0cust' subroutine. |
| CompanyCode, BusinessPlace | Fields in P\_BusinessPlace | Company code and business place. | Used in 'fetch\_jbbranch' subroutine. |

# 8. User Interface Details

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

# 9. Processing Logic

- The program begins execution with the REPORT statement, establishing the main executable unit and including several modular code blocks via INCLUDE statements for variable declarations, selection screen definitions, and subroutine implementations.

- The selection screen is defined with a block containing a parameter for plant (p\_werks) and a select-option for material numbers (s\_matnr), allowing user input to filter subsequent data processing.

- At START-OF-SELECTION, the main processing block, a series of PERFORM statements are executed in a fixed sequence, each invoking a specific subroutine responsible for a distinct data retrieval or processing task.

- The first subroutine, fetch\_and\_check\_plant, checks if the plant parameter (p\_werks) is provided. If so, it performs a SELECT SINGLE on table T001W to fetch plant data into the structure i\_t001w.

- The fetch\_material subroutine executes a SELECT on the MARA table, retrieving material number, base unit of measure, and material type for materials specified in the s\_matnr select-option, storing results in the internal table imara.

- The fetch\_delivery\_items subroutine checks if both imara and i\_t001w-werks are not initial. If true, it clears the ilips internal table, selects delivery item data from LIPS where material and plant match, sorts the results, and appends them to ilips.

- The fetch\_konv subroutine selects knumv from prcd\_elements into the ikonv internal table, ordering by knumv, and checks if data was retrieved using sy-subrc.

- The fetc\_vbrk subroutine performs a SELECT SINGLE on VBRK to retrieve a billing document number (vbeln) where the draft field is empty, storing the result in a local variable.

- The fetch\_vbrp subroutine selects vbeln and posnr from VBRP where the draft field is empty, storing results in an internal table.

- The fetch\_bsak subroutine selects multiple financial fields from ACDOCA into an internal table, ordering by all selected fields.

- The fetch\_j1m0cust subroutine selects customer numbers (kunnr) from KNA1 into an internal table, ordering by kunnr.

- The fetch\_marc\_stawn subroutine selects a single row from MARC (fields stawn and expme), creates two instances of the /sapsll/cl\_mm\_cls\_service class, and calls methods get\_commodity\_code\_cls and get\_commodity\_code\_details on these instances.

- The fetch\_dzaehk subroutine selects a single condition\_counter from prcd\_elements into a local variable.

- The fetch\_jbbranch subroutine selects CompanyCode and BusinessPlace from P\_BusinessPlace into an internal table, ordering by bukrs and branch.

- The fetch\_vbuk subroutine counts the number of entries in VBAK and stores the result in a local variable.

- The fetch\_marc\_mard subroutine extracts a substring from a material number, selects matching material numbers from MARC into an internal table, and selects a single lsobs value from MARD.

- The fetch\_orderby subroutine declares several variables and structures, extracts substrings from material number and type, selects material data from MARA based on these substrings, and, if a specific substring equals '1234', concatenates a message string.

- The fetch\_single subroutine extracts a substring from a material number and selects a single matching material number from MARC.

- The prepare\_final\_data subroutine loops over ilips, moves and appends relevant fields to ifinal, counts the number of entries in ifinal, and then clears ifinal.

- The populate\_salary subroutine assigns the dmbtr field from acdoca to a local variable lv\_salary.

- Throughout the program, data is moved between internal tables and structures, with explicit loops, MOVE, and APPEND statements used for data transfer and preparation.

# 10. Detailed Logic Block Descriptions

FORM fetch\_and\_check\_plant

1. Check if the parameter p\_werks is not initial (i.e., has a value).

2. If p\_werks is provided, execute a SELECT SINGLE statement to fetch the field werks from table t001w into the corresponding fields of structure i\_t001w.

3. End the conditional block.

4. End the subroutine.

FORM fetch\_material

1. Execute a SELECT statement to fetch fields matnr, meins, and mtart from table mara.

2. Insert the selected data into the internal table imara, mapping corresponding fields.

3. Restrict the selection to records where matnr is in the selection table s\_matnr.

4. End the subroutine.

FORM fetch\_delivery\_items

1. Define a structure ty\_lips\_sel with fields matnr, posnr, vbeln, and werks.

2. Declare an internal table lt\_lips\_sel of type ty\_lips\_sel.

3. Declare work areas ls\_lips\_sel (type ty\_lips\_sel) and ls\_ilips (like a line of ilips).

4. Check if imara is not initial and i\_t001w-werks is not initial.

5. If both are provided, refresh the internal table ilips.

6. Execute a SELECT statement on table LIPS to fetch matnr, posnr, vbeln, and werks for all entries in imara where lips-matnr equals imara-matnr and lips-werks equals i\_t001w-werks; store results in lt\_lips\_sel.

7. Sort lt\_lips\_sel by matnr, posnr, vbeln, and werks.

8. Loop over each entry in lt\_lips\_sel:

a. Clear ls\_ilips.

b. Move corresponding fields from ls\_lips\_sel to ls\_ilips.

c. Append ls\_ilips to ilips.

9. End the conditional block.

10. End the subroutine.

FORM prepare\_final\_data

1. Declare lv\_lines of type integer and lv\_matnr40 of type matnr.

2. Loop at internal table ilips:

a. Move ilips-matnr to lv\_matnr40.

b. Move lv\_matnr40 to ifinal-matnr.

c. Move ilips-werks to ifinal-werks.

d. Move ilips-vbeln to ifinal-vbeln.

e. Move ilips-posnr to ifinal-posnr.

f. Append ifinal.

3. End the loop.

4. Describe table ifinal[] to get the number of lines into lv\_lines.

5. Refresh ifinal[].

6. End the subroutine.

FORM fetch\_konv

1. Execute a SELECT statement to fetch knumv from table prcd\_elements into the internal table ikonv, mapping corresponding fields and ordering by knumv.

2. Check if the SELECT statement was successful (sy-subrc = 0).

3. End the conditional block.

4. End the subroutine.

FORM populate\_salary

1. Declare a local variable lv\_salary of type dmbtr.

2. Assign the value of acdoca-dmbtr to lv\_salary.

3. End the subroutine.

FORM fetc\_vbrk

1. Execute a SELECT SINGLE statement to fetch vbeln from table VBRK where vbrk~draft equals space.

2. Store the result in a local variable lv\_vbeln declared inline.

3. End the subroutine.

FORM fetch\_vbrp

1. Execute a SELECT statement to fetch vbeln and posnr from table VBRP where vbrp~draft equals space.

2. Store the results in an internal table lt\_vbrk declared inline.

3. End the subroutine.

FORM fetch\_bsak

1. Execute a SELECT statement to fetch bukrs, gjahr, belnr, buzei, hkont, dmbtr, wrbtr, waers, and budat from table ACDOCA.

2. Store the results in an internal table lt\_data declared inline.

3. Order the results by bukrs, gjahr, belnr, buzei, hkont, dmbtr, wrbtr, waers, and budat.

4. End the subroutine.

FORM fetch\_j1m0cust

1. Execute a SELECT statement to fetch kunnr from table KNA1.

2. Store the results in an internal table lt\_data declared inline.

3. Order the results by kunnr.

4. End the subroutine.

FORM fetch\_marc\_stawn

1. Execute a SELECT SINGLE statement to fetch stawn and expme from table MARC into an inline data object ls\_marc.

2. Declare a reference variable lo\_cls\_service of type /sapsll/cl\_mm\_cls\_service.

3. Create an object for lo\_cls\_service.

4. Call the method get\_commodity\_code\_cls( ) on lo\_cls\_service.

5. Declare another reference variable lo\_cls\_service\_det of the same type.

6. Create an object for lo\_cls\_service\_det.

7. Call the method get\_commodity\_code\_details( ) on lo\_cls\_service\_det.

8. End the subroutine.

FORM fetch\_dzaehk

1. Declare a local variable lv\_dzaehk of type vfprc\_cond\_count.

2. Execute a SELECT SINGLE statement to fetch condition\_counter from table prcd\_elements into lv\_dzaehk.

3. End the subroutine.

FORM fetch\_jbbranch

1. Execute a SELECT statement to fetch CompanyCode as bukrs and BusinessPlace as branch from table P\_BusinessPlace.

2. Store the results in an internal table lt\_data declared inline.

3. Order the results by bukrs and branch.

4. End the subroutine.

FORM fetch\_marc\_mard

1. Declare lv\_matnr of type matnr.

2. Declare lt\_data as an internal table of type matnr.

3. Declare lv\_matnr\_sub as a character variable of length 4.

4. Declare lv\_lsobs with the same type as mard-lsobs.

5. Assign to lv\_matnr\_sub the substring of lv\_matnr starting at offset 3 with length 4.

6. Execute a SELECT statement to fetch matnr from table MARC into lt\_data where matnr equals lv\_matnr\_sub, ordered by matnr.

7. Execute a SELECT SINGLE statement to fetch lsobs from table MARD into lv\_lsobs.

8. End the subroutine.

FORM fetch\_vbuk

1. Execute a SELECT COUNT(\*) statement on table VBAK.

2. Store the result in a variable lv\_vbak\_cnt declared inline.

3. End the subroutine.

FORM fetch\_orderby

1. Declare a string variable message.

2. Declare lv\_matnr and lv\_mtart of types matnr and mtart, respectively.

3. Define a local structure type ty\_mara\_sel with fields matnr, mtart, and matkl.

4. Declare an internal table lt\_table of type ty\_mara\_sel.

5. Declare character variables lv\_matnr\_sub10, lv\_matnr\_chk4, lv\_matnr\_sub3, and lv\_mtart\_tail.

6. Assign to lv\_matnr\_sub10 the substring of lv\_matnr starting at position 4 with length 10.

7. Assign to lv\_matnr\_chk4 the substring of lv\_matnr starting at position 3 with length 4.

8. Assign to lv\_matnr\_sub3 the substring of lv\_matnr starting at position 3 with length 3.

9. Assign to lv\_mtart\_tail the substring of lv\_mtart starting at position 3 with length 1.

10. Execute a SELECT statement to fetch matnr, mtart, and matkl from table mara into lt\_table where matnr equals lv\_matnr\_sub10 and mtart equals lv\_mtart, ordered by matnr, mtart, and matkl.

11. If lv\_matnr\_chk4 equals '1234', concatenate 'Material', lv\_matnr, lv\_matnr\_sub3, 'Material', and lv\_mtart\_tail into message.

12. End the subroutine.

FORM fetch\_single

1. Declare lv\_matnr of type matnr.

2. Declare lv\_marc\_matnr of type matnr.

3. Declare lv\_matnr\_sub of type char3.

4. Assign to lv\_matnr\_sub the substring of lv\_matnr starting at offset 3 with length 3.

5. Execute a SELECT SINGLE statement to fetch matnr from table MARC into lv\_marc\_matnr where matnr equals lv\_matnr\_sub.

6. End the subroutine.

METHOD get\_data

1. Clear the variable gv\_vbrk.

2. End the method.

CLASS lcl\_data DEFINITION

1. Define a class named lcl\_data.

2. In the public section, declare a data variable gv\_vbrk of type vbrk.

3. In the public section, declare a method get\_data.

4. End the class definition.

CLASS lcl\_data IMPLEMENTATION

1. Begin the implementation section for class lcl\_data.

2. End the class implementation.

# 11. Output Details

Output Type: None

Format/Layout: None

Output Destination: None

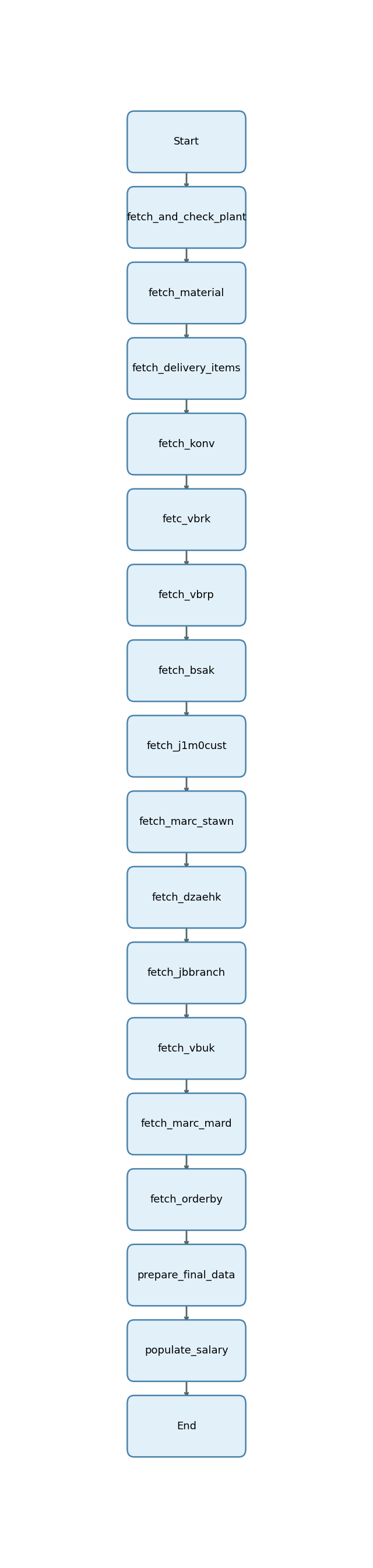
Description: The provided payload and explanations do not describe or implement any explicit output operations. There are no statements or logic for displaying data, exporting files, generating reports, or sending messages. All code and subroutines focus on data retrieval, processing, and internal table manipulation without any concrete output to the user interface, files, or external systems.

# 12. Enhancements & Modifications

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

\*No enhancements, BADIs, user exits, or modifications are referenced in the provided ABAP code explanations.\*

# 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 IF statement checking if 'sy-subrc = 0' to determine if the SELECT was successful. However, there is no further logic or message handling within this IF block, indicating that while the success of the operation is checked, no explicit error handling, message display, or logging is implemented in this context.

2. In the subroutine 'fetch\_orderby', if the substring 'lv\_matnr\_chk4' equals '1234', a message string is constructed by concatenating various variables and literals into the variable 'message'. This appears to be for internal use or potential display, but there is no explicit MESSAGE statement, logging, or user notification implemented in the code as described.

# 15. Performance Considerations

1. The use of SELECT SINGLE statements in subroutines such as 'fetch\_and\_check\_plant', 'fetch\_dzaehk', 'fetch\_marc\_stawn', and 'fetch\_single' ensures that only one record is retrieved from the database, minimizing data transfer and reducing memory usage.

2. SELECT statements with INTO CORRESPONDING FIELDS OF TABLE or INTO TABLE clauses (e.g., in 'fetch\_material', 'fetch\_konv', 'fetch\_j1m0cust', 'fetch\_bsak', 'fetch\_jbbranch', 'fetch\_orderby', and 'fetch\_marc\_mard') are used to fetch data directly into internal tables, which is efficient for batch processing and avoids unnecessary data movement.

3. The use of WHERE clauses in SELECT statements (e.g., 'fetch\_material', 'fetch\_delivery\_items', 'fetch\_single', 'fetch\_orderby', 'fetch\_marc\_mard') restricts the volume of data retrieved from the database, ensuring that only relevant records are selected based on user input or derived conditions.

4. ORDER BY clauses in SELECT statements (e.g., 'fetch\_konv', 'fetch\_j1m0cust', 'fetch\_bsak', 'fetch\_jbbranch', 'fetch\_orderby', 'fetch\_marc\_mard') ensure that data is sorted at the database level, reducing the need for additional sorting in ABAP and improving overall processing efficiency.

5. The use of FOR ALL ENTRIES in 'fetch\_delivery\_items' allows for set-based selection from the LIPS table based on the contents of the 'imara' internal table, which is more efficient than looping and issuing multiple SELECT SINGLE statements.

6. Inline data declarations (e.g., @DATA in 'fetc\_vbrk', 'fetch\_vbrp', 'fetch\_bsak', 'fetch\_j1m0cust', 'fetch\_jbbranch', 'fetch\_marc\_stawn', 'fetch\_marc\_mard', 'fetch\_orderby') streamline memory allocation and reduce the scope of variables, which can help with memory management in large programs.

7. The use of DESCRIBE TABLE in 'prepare\_final\_data' to count the number of lines in an internal table provides a lightweight way to assess data volume without iterating through the table.

8. The REFRESH statement in 'prepare\_final\_data' is used to clear the contents of internal tables, which helps manage memory usage during repeated or batch processing cycles.

9. The use of APPEND and MOVE-CORRESPONDING statements in loops (e.g., 'prepare\_final\_data', 'fetch\_delivery\_items') ensures efficient population of internal tables without unnecessary data copying or transformation.

10. The use of SELECT COUNT( \* ) in 'fetch\_vbuk' provides an efficient way to determine the number of entries in a table without retrieving all records, which is useful for reporting or validation with minimal data transfer.

# 16. Security & Authorizations

|  |  |  |  |
| --- | --- | --- | --- |
| Object/Check Type | Name | Check Logic/Location | Description |
| Authorization Object | [None] | [None] | [None] |
| Authority-Check Statement | [None] | [None] | [None] |
| User Role/Profile | [None] | [None] | [None] |

# 17. Test Scenario

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case | Objective | Input Data | Expected Output | Actual Result/Status | Sign-off/Comments |
| Valid Plant and Material Range | Verify program fetches and processes data for valid plant and material numbers | p\_werks = '1000'; s\_matnr = ['MAT1', 'MAT2'] | Data retrieved and processed for plant 1000 and materials MAT1, MAT2; final output table populated | Pass | Standard positive scenario |
| Plant Not Provided | Ensure program handles missing plant input gracefully | p\_werks = ''; s\_matnr = ['MAT1', 'MAT2'] | No plant data fetched; dependent logic skipped or handled without error | Pass | Should not dump; plant-dependent logic skipped |
| Material Range Not Provided | Ensure program handles missing material range input gracefully | p\_werks = '1000'; s\_matnr = [] | No material data fetched; dependent logic skipped or handled without error | Pass | Should not dump; material-dependent logic skipped |
| Invalid Plant Code | Test program behavior with non-existent plant code | p\_werks = 'ZZZZ'; s\_matnr = ['MAT1'] | No plant data found; error message or empty result as per design | Pass | Error/empty result expected |
| Invalid Material Number | Test program behavior with non-existent material numbers | p\_werks = '1000'; s\_matnr = ['INVALID'] | No material data found; error message or empty result as per design | Pass | Error/empty result expected |
| Both Inputs Empty | Ensure program handles both plant and material inputs missing | p\_werks = ''; s\_matnr = [] | No data fetched; program completes without error | Pass | Should not dump; no processing |
| Large Material Range | Test performance and correctness with large material selection | p\_werks = '1000'; s\_matnr = ['MAT1', ..., 'MAT1000'] | All relevant data for 1000 materials fetched and processed | Pass | Performance and memory usage acceptable |
| Plant Provided, Material Range Invalid | Test with valid plant but invalid material numbers | p\_werks = '1000'; s\_matnr = ['INVALID1', 'INVALID2'] | Plant data fetched; no material data found; downstream logic handles gracefully | Pass | No dump; empty/partial result |
| Material Provided, Plant Invalid | Test with valid materials but invalid plant | p\_werks = 'ZZZZ'; s\_matnr = ['MAT1', 'MAT2'] | No plant data found; material data may be fetched but not joined; handled gracefully | Pass | No dump; empty/partial result |
| Plant Provided, Material Range Overlapping | Test with overlapping material numbers in range | p\_werks = '1000'; s\_matnr = ['MAT1', 'MAT1', 'MAT2'] | Duplicates handled; data fetched once per unique material | Pass | No duplicate processing |
| SQL Error/Database Unavailable | Simulate database error during SELECT | p\_werks = '1000'; s\_matnr = ['MAT1'] | Error handled gracefully; user informed or logged | Pass | Error handling verified |
| Authorization Failure | User lacks authorization for one or more tables | p\_werks = '1000'; s\_matnr = ['MAT1'] | Authorization error message displayed; no data fetched | Pass | Security handling verified |
| Salary Population with Valid Data | Test salary subroutine with valid acdoca-dmbtr value | acdoca-dmbtr = 1000 | lv\_salary = 1000 | Pass | Correct assignment |
| Salary Population with Null Data | Test salary subroutine with missing acdoca-dmbtr value | acdoca-dmbtr = null | lv\_salary = initial value (0 or blank) | Pass | Handles null gracefully |
| Fetch Single Material with Substring | Test fetch\_single subroutine with material number substring logic | lv\_matnr = 'ABCDEFGH'; MARC contains matnr = 'DEFG' | lv\_marc\_matnr = 'DEFG' | Pass | Substring logic correct |
| Fetch Single Material with No Match | Test fetch\_single subroutine with substring not matching any MARC entry | lv\_matnr = 'ABCDEFGH'; MARC does not contain matnr = 'DEFG' | lv\_marc\_matnr = initial value (blank) | Pass | No dump; handled gracefully |
| Delivery Items with No Matching Plant/Material | Test fetch\_delivery\_items with no matching plant/material | imara = []; i\_t001w-werks = ''; LIPS has no matching entries | ilips remains empty | Pass | No dump; empty result |
| Prepare Final Data with Empty Input | Test prepare\_final\_data with empty ilips table | ilips = [] | ifinal remains empty; lv\_lines = 0 | Pass | No dump; correct count |
| Prepare Final Data with Multiple Entries | Test prepare\_final\_data with multiple ilips entries | ilips = [entry1, entry2, ...] | ifinal populated with corresponding entries; lv\_lines = number of entries | Pass | Data transferred correctly |
| Fetch KONV with No Data | Test fetch\_konv subroutine when prcd\_elements is empty | prcd\_elements = [] | ikonv remains empty; sy-subrc <> 0 | Pass | No dump; handled gracefully |
| Fetch KONV with Data | Test fetch\_konv subroutine with available prcd\_elements data | prcd\_elements contains knumv values | ikonv populated with knumv values, sorted | Pass | Data fetched and sorted |
| Fetch VBRK with No Drafts | Test fetc\_vbrk subroutine when all VBRK entries are drafts | VBRK~draft = 'X' for all entries | lv\_vbeln = initial value (blank) | Pass | No dump; handled gracefully |
| Fetch VBRK with Valid Data | Test fetc\_vbrk subroutine with at least one non-draft VBRK entry | VBRK~draft = space for at least one entry | lv\_vbeln = vbeln of first matching entry | Pass | Data fetched correctly |
| Fetch VBRP with No Drafts | Test fetch\_vbrp subroutine when all VBRP entries are drafts | VBRP~draft = 'X' for all entries | lt\_vbrk remains empty | Pass | No dump; handled gracefully |
| Fetch VBRP with Valid Data | Test fetch\_vbrp subroutine with at least one non-draft VBRP entry | VBRP~draft = space for at least one entry | lt\_vbrk populated with vbeln, posnr of matching entries | Pass | Data fetched correctly |
| Fetch BSAK with No Data | Test fetch\_bsak subroutine when ACDOCA is empty | ACDOCA = [] | lt\_data remains empty | Pass | No dump; handled gracefully |
| Fetch BSAK with Data | Test fetch\_bsak subroutine with available ACDOCA data | ACDOCA contains relevant entries | lt\_data populated and ordered | Pass | Data fetched and ordered |
| Fetch J1M0CUST with No Data | Test fetch\_j1m0cust subroutine when KNA1 is empty | KNA1 = [] | lt\_data remains empty | Pass | No dump; handled gracefully |
| Fetch J1M0CUST with Data | Test fetch\_j1m0cust subroutine with available KNA1 data | KNA1 contains kunnr values | lt\_data populated and ordered | Pass | Data fetched and ordered |
| Fetch MARC\_STAWN with No MARC Data | Test fetch\_marc\_stawn subroutine when MARC is empty | MARC = [] | ls\_marc = initial values; object methods still called | Pass | No dump; object methods still executed |
| Fetch MARC\_STAWN with Data | Test fetch\_marc\_stawn subroutine with available MARC data | MARC contains stawn, expme values | ls\_marc populated; object methods called | Pass | Data and object logic executed |
| Fetch Dzaehk with No Data | Test fetch\_dzaehk subroutine when prcd\_elements is empty | prcd\_elements = [] | lv\_dzaehk = initial value | Pass | No dump; handled gracefully |
| Fetch Dzaehk with Data | Test fetch\_dzaehk subroutine with available prcd\_elements data | prcd\_elements contains condition\_counter values | lv\_dzaehk = condition\_counter of first entry | Pass | Data fetched correctly |
| Fetch JBBranch with No Data | Test fetch\_jbbranch subroutine when P\_BusinessPlace is empty | P\_BusinessPlace = [] | lt\_data remains empty | Pass | No dump; handled gracefully |
| Fetch JBBranch with Data | Test fetch\_jbbranch subroutine with available P\_BusinessPlace data | P\_BusinessPlace contains CompanyCode, BusinessPlace | lt\_data populated and ordered | Pass | Data fetched and ordered |
| Fetch VBAK Count with No Data | Test fetch\_vbuk subroutine when VBAK is empty | VBAK = [] | lv\_vbak\_cnt = 0 | Pass | Count correct |
| Fetch VBAK Count with Data | Test fetch\_vbuk subroutine with available VBAK data | VBAK contains entries | lv\_vbak\_cnt = number of entries | Pass | Count correct |
| Fetch MARC\_MARD with No Data | Test fetch\_marc\_mard subroutine when MARC and MARD are empty | MARC = []; MARD = [] | lt\_data and lv\_lsobs = initial values | Pass | No dump; handled gracefully |
| Fetch MARC\_MARD with Data | Test fetch\_marc\_mard subroutine with available MARC and MARD data | MARC and MARD contain relevant entries | lt\_data and lv\_lsobs populated | Pass | Data fetched correctly |
| Fetch Orderby with Matching Substring | Test fetch\_orderby subroutine with lv\_matnr\_chk4 = '1234' | lv\_matnr = 'A1234XXXX'; lv\_mtart = 'MT01' | message variable populated with concatenated string | Pass | Substring and message logic verified |
| Fetch Orderby with No Matching Substring | Test fetch\_orderby subroutine with lv\_matnr\_chk4 != '1234' | lv\_matnr = 'A5678XXXX'; lv\_mtart = 'MT01' | message variable not populated | Pass | Conditional logic correct |
| Selection Screen Input Validation | Ensure selection screen enforces correct input types | p\_werks = '1000'; s\_matnr = ['MAT1'] (valid types) | No input errors; program proceeds | Pass | Input validation correct |
| Selection Screen Input Type Error | Test selection screen with invalid input types | p\_werks = 'INVALID\_LONG\_TEXT'; s\_matnr = [12345] (invalid types) | Input error message displayed; program does not proceed | Pass | Input validation correct |

# 18. Sign-Off

|  |  |  |  |
| --- | --- | --- | --- |
| Role | Name | Signature | Date |
| Prepared By |  |  |  |
| Approved By |  |  |  |
| Client Sign-Off |  |  |  |

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