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 for ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 | [To be filled] | [To be filled] | PWC AI Asset | [To be filled] |

# 2. Introduction

This document provides the technical specification for the SAP ABAP program ZRCOPY\_SAMPLE\_ECC\_CODE\_V1. The objective is to outline the design, structure, and implementation details of the program, which is developed as a modular ABAP report utilizing includes, subroutines, and object-oriented components. The scope covers the program's data retrieval, processing logic, and integration with SAP standard tables for materials, deliveries, billing, and financial data. The intended audience includes SAP ABAP developers, technical consultants, and project stakeholders involved in the development, review, or maintenance of this solution.

# 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 is to develop an ABAP program that efficiently retrieves, processes, and prepares data from various SAP standard tables such as materials, deliveries, billing documents, and financial records. The solution aims to provide a structured and automated approach for extracting relevant information based on user input (such as plant and material numbers), performing necessary validations and data transformations, and preparing the final dataset for reporting or further processing. The objective is to streamline data handling processes, ensure data integrity, and support business operations with accurate and timely information.

# 5. Solution Approach

1. Modular Program Structure: The solution is architected using a main ABAP report with multiple includes, separating selection screen logic, data declarations, and processing routines. This modular approach enhances maintainability and reusability.

2. Comprehensive Data Retrieval: The program leverages a series of PERFORM subroutines to fetch and process data from key SAP tables (such as MARA, LIPS, VBRK, VBRP, ACDOCA, KNA1, MARC, MARD, and others) based on user-provided selection criteria, ensuring targeted and efficient data extraction.

3. Object-Oriented and Procedural Integration: The design incorporates both object-oriented ABAP (local class lcl\_data with methods) and traditional procedural techniques (FORM routines), allowing for encapsulation of logic where appropriate and adherence to SAP best practices.

# 6. SAP Object Details

|  |  |  |  |
| --- | --- | --- | --- |
| Object Type | Object Name | Description | Related Main Program/Module |
| Program | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 | Main ABAP report program for data retrieval 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 global variables | 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 data | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_and\_check\_plant | Subroutine to fetch and validate plant data | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_material | Subroutine to fetch material master data | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_delivery\_items | Subroutine to fetch delivery item data | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_konv | Subroutine to fetch condition records from prcd\_elements | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetc\_vbrk | Subroutine to fetch billing document header data | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_vbrp | Subroutine to fetch billing document item data | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_bsak | Subroutine to fetch financial document data from ACDOCA | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_j1m0cust | Subroutine to fetch customer master data | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_marc\_stawn | Subroutine to fetch MARC table data and commodity code classification | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_dzaehk | Subroutine to fetch condition counter from prcd\_elements | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_jbbranch | Subroutine to fetch business place data | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_vbuk | Subroutine to count sales document headers in VBAK | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_marc\_mard | Subroutine to fetch material and storage location data from MARC and MARD | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_orderby | Subroutine to fetch and process material data with ordering and message construction | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_single | Subroutine to fetch a single material number from MARC based on substring logic | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | prepare\_final\_data | Subroutine to prepare the final output dataset | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | populate\_salary | Subroutine to populate salary data from ACDOCA | 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, fetched from table PRCD\_ELEMENTS (knumv field), for pricing data processing. |
| 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 combining delivery and material information for output or further use. |
| 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 fetched from T001W table for plant validation and processing. |
| t001w | t001w | Table work area declaration for database table T001W. | Used as a work area for plant data during SELECT and processing operations. |
| mara | mara | Table work area declaration for database table MARA. | Used as a work area for material master data during SELECT and processing operations. |
| lips | lips | Table work area declaration for database table LIPS. | Used as a work area for delivery item data during SELECT and processing operations. |
| bseg | bseg | Table work area declaration for database table BSEG. | Used as a work area for accounting document segment data. |
| ilips | OCCURS 0 structure based on LIPS | Internal table with header line based on structure LIPS. | Used to store delivery item data fetched from LIPS table for further processing. |
| imara | OCCURS 0 structure based on MARA | Internal table with header line based on structure MARA. | Used to store material master data fetched from MARA table for further processing. |
| lv\_product | TYPE matnr | Variable of type MATNR for storing a product number. | Used to temporarily store material number during processing. |
| lv\_salary | TYPE dmbtr | Variable of type DMBTR for storing a salary amount. | Used to store salary/amount values, e.g., from ACDOCA-DMBTR. |
| 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. |
| lt\_lips\_sel | STANDARD TABLE OF ty\_lips\_sel | Internal table for selected delivery item data (matnr, posnr, vbeln, werks). | Used in fetch\_delivery\_items subroutine to temporarily store selected LIPS data. |
| ls\_lips\_sel | ty\_lips\_sel | Work area for processing a single row of lt\_lips\_sel. | Used in fetch\_delivery\_items subroutine for row-wise processing. |
| ls\_ilips | LIKE LINE OF ilips | Work area for a single line of ilips internal table. | Used in fetch\_delivery\_items subroutine for data transfer. |
| ty\_lips\_sel | STRUCTURE (matnr, posnr, vbeln, werks) | Local structure type for selected LIPS fields. | Used in fetch\_delivery\_items subroutine for typed data selection. |
| lv\_lines | TYPE i | Integer variable for counting lines in an internal table. | Used in prepare\_final\_data subroutine to store number of lines in ifinal. |
| lv\_matnr40 | TYPE matnr | Variable for storing material number (40 characters). | Used in prepare\_final\_data subroutine for temporary storage. |
| lv\_vbeln | TYPE vbeln (inline @DATA) | Variable for storing billing document number. | Used in fetc\_vbrk subroutine to store result of SELECT SINGLE from VBRK. |
| lt\_vbrk | TABLE (inline @DATA) | Internal table for storing billing document items. | Used in fetch\_vbrp subroutine to store results from VBRP table. |
| lt\_data | TABLE (inline @DATA) | Internal table for storing generic data from SELECT statements. | Used in fetch\_bsak, fetch\_j1m0cust, fetch\_jbbranch, fetch\_marc\_mard subroutines. |
| ls\_marc | STRUCTURE (inline @DATA) | Work area for storing MARC table fields (stawn, expme). | Used in fetch\_marc\_stawn subroutine for SELECT SINGLE from MARC. |
| lo\_cls\_service | TYPE REF TO /sapsll/cl\_mm\_cls\_service | Reference variable for commodity code classification service class. | Used in fetch\_marc\_stawn subroutine for calling get\_commodity\_code\_cls method. |
| lo\_cls\_service\_det | TYPE REF TO /sapsll/cl\_mm\_cls\_service | Reference variable for commodity code classification service class (details). | Used in fetch\_marc\_stawn subroutine for calling get\_commodity\_code\_details method. |
| lv\_dzaehk | TYPE vfprc\_cond\_count | Variable for storing condition counter. | Used in fetch\_dzaehk subroutine to store result from PRCD\_ELEMENTS. |
| lv\_matnr | TYPE matnr | Variable for storing material number. | Used in fetch\_marc\_mard and fetch\_single subroutines. |
| lv\_matnr\_sub | TYPE c LENGTH 4 / char3 | Variable for storing substring of material number. | Used in fetch\_marc\_mard and fetch\_single subroutines for substring operations. |
| lv\_lsobs | LIKE mard-lsobs | Variable for storing LSOBS field from MARD table. | Used in fetch\_marc\_mard subroutine. |
| lv\_marc\_matnr | TYPE matnr | Variable for storing material number from MARC. | Used in fetch\_single subroutine. |
| message | TYPE string | Variable for storing concatenated message. | Used in fetch\_orderby subroutine for message construction. |
| lv\_mtart | TYPE mtart | Variable for storing material type. | Used in fetch\_orderby subroutine. |
| ty\_mara\_sel | STRUCTURE (matnr, mtart, matkl) | Local structure type for selected MARA fields. | Used in fetch\_orderby subroutine for typed data selection. |
| lt\_table | STANDARD TABLE OF ty\_mara\_sel | Internal table for storing selected MARA 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 for checking. | 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. |
| gv\_vbrk | TYPE vbrk | Variable for storing billing document header data. | Used in lcl\_data class and get\_data method. |
| SAP Table: T001W | Database Table | Plant master data table. | Read in fetch\_and\_check\_plant subroutine and for plant validation. |
| SAP Table: MARA | Database Table | Material master data table. | Read in fetch\_material, fetch\_orderby, and for material master processing. |
| SAP Table: LIPS | Database Table | Delivery item data table. | Read in fetch\_delivery\_items subroutine for delivery item processing. |
| SAP Table: BSEG | Database Table | Accounting document segment table. | Used as work area for accounting data. |
| SAP Table: PRCD\_ELEMENTS | Database Table | Pricing condition elements table. | Read in fetch\_konv and fetch\_dzaehk subroutines for condition records. |
| SAP Table: VBRK | Database Table | Billing document header table. | Read in fetc\_vbrk subroutine for billing document number. |
| SAP Table: VBRP | Database Table | Billing document item table. | Read in fetch\_vbrp subroutine for billing document items. |
| SAP Table: ACDOCA | Database Table | Universal journal entry line items. | Read in fetch\_bsak and populate\_salary subroutines for financial data. |
| SAP Table: KNA1 | Database Table | Customer master table. | Read in fetch\_j1m0cust subroutine for customer data. |
| SAP Table: MARC | Database Table | Plant data for material. | Read in fetch\_marc\_stawn, fetch\_marc\_mard, and fetch\_single subroutines. |
| SAP Table: MARD | Database Table | Storage location data for material. | Read in fetch\_marc\_mard subroutine for storage location data. |
| SAP Table: VBAK | Database Table | Sales document header data. | Read in fetch\_vbuk subroutine for counting sales documents. |
| SAP Table: P\_BusinessPlace | Database Table | Business place data table. | Read in fetch\_jbbranch subroutine for company code and business place data. |

# 8. User Interface Details

|  |  |  |  |
| --- | --- | --- | --- |
| Screen Field | Type | Default Value | Description |
| p\_werks | Parameter |  | Single-value input field for plant (WERKS) based on table T001W. |
| s\_matnr | Select-Option |  | Range input for material number (MATNR) based on table MARA. |

# 9. Processing Logic

- The program begins execution with the REPORT statement, establishing 'ZRCOPY\_SAMPLE\_ECC\_CODE\_V1' as the main executable unit.

- Several INCLUDE statements are used to modularize the code, bringing in variable declarations, selection screen definitions, and subroutine implementations from separate includes.

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

- The START-OF-SELECTION event block serves as the main processing section, where a series of PERFORM statements are executed in sequence, each invoking a specific 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 internal table i\_t001w.

- fetch\_material: Executes a SELECT on MARA to retrieve material numbers, base unit of measure, and material type for materials specified in s\_matnr, storing results in internal table imara.

- fetch\_delivery\_items: If imara and i\_t001w-werks are not initial, clears ilips, selects delivery item data (matnr, posnr, vbeln, werks) from LIPS for materials in imara and the specified plant, sorts the results, and appends them to ilips.

- fetch\_konv: Selects knumv from prcd\_elements into ikonv, ordered by knumv, and checks if data was fetched successfully.

- fetc\_vbrk: Selects a single vbeln from VBRK where draft is initial (empty), storing the result in a local variable.

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

- fetch\_bsak: Selects multiple financial fields from ACDOCA into an internal table, ordered by all selected fields.

- fetch\_j1m0cust: Selects kunnr from KNA1 into an internal table, ordered by kunnr.

- fetch\_marc\_stawn: Selects a single stawn and expme from MARC, creates two instances of /sapsll/cl\_mm\_cls\_service, and calls methods get\_commodity\_code\_cls and get\_commodity\_code\_details on them.

- fetch\_dzaehk: Selects a single condition\_counter from prcd\_elements into a local variable.

- fetch\_jbbranch: Selects CompanyCode and BusinessPlace from P\_BusinessPlace into an internal table, ordered by bukrs and branch.

- fetch\_vbuk: Counts the number of entries in VBAK and stores the result in a local variable.

- fetch\_marc\_mard: Extracts a substring from a material number, selects matching matnr from MARC into an internal table, and selects a single lsobs from MARD into a variable.

- fetch\_orderby: Declares several variables and structures, extracts substrings from material number and type, selects matnr, mtart, and matkl from MARA into an internal table based on substring conditions, and constructs a message if a specific substring matches '1234'.

- prepare\_final\_data: Loops over ilips, moves fields to ifinal, appends to ifinal, counts the number of entries, and then refreshes ifinal.

- populate\_salary: Declares a local salary variable, assigns the value of acdoca-dmbtr to it.

- Data declarations are made for internal tables and variables, including ilips (structure of LIPS), imara (structure of MARA), ikonv (structure of KONV), ifinal (custom structure), and i\_t001w (structure of T001W with header line).

- The program uses explicit loops (LOOP AT ilips) and conditional checks (IF NOT p\_werks IS INITIAL, IF imara IS NOT INITIAL AND i\_t001w-werks IS NOT INITIAL, IF lv\_matnr\_chk4 = '1234') to control the flow of data processing.

- Class lcl\_data is defined with a public data variable gv\_vbrk and a method get\_data, whose implementation simply clears gv\_vbrk.

- The overall flow is strictly sequential, with each subroutine performing a specific data retrieval or processing task, and the results being stored in internal tables or variables for further use or output.

# 10. Detailed Logic Block Descriptions

fetch\_and\_check\_plant

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

2. If p\_werks has a value, execute a SELECT SINGLE statement to retrieve the field werks from the t001w table.

3. Store the result into the corresponding fields of the structure i\_t001w.

4. End the conditional block.

fetch\_material

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

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

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

4. End the subroutine.

fetch\_delivery\_items

1. Define a local 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 not initial, refresh the internal table ilips.

6. Execute a SELECT statement on the LIPS table 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.

7. Store the results in lt\_lips\_sel.

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

9. 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.

10. End the conditional block.

prepare\_final\_data

1. Declare lv\_lines as an integer and lv\_matnr40 as type matnr.

2. Loop at 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 the table ifinal[] to get the number of lines into lv\_lines.

5. Refresh ifinal[].

fetch\_konv

1. Execute a SELECT statement to retrieve knumv from prcd\_elements.

2. Insert the results into the internal table ikonv, mapping corresponding fields, and order by knumv.

3. Check if sy-subrc equals 0 (indicating successful selection).

4. End the conditional block.

populate\_salary

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

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

fetc\_vbrk

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

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

fetch\_vbrp

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

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

fetch\_bsak

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

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.

fetch\_j1m0cust

1. Execute a SELECT statement to retrieve kunnr from the KNA1 table.

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

3. Order the results by kunnr.

fetch\_marc\_stawn

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

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

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

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

fetch\_dzaehk

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

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

fetch\_jbbranch

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

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

3. Order the results by bukrs and branch.

fetch\_marc\_mard

1. Declare lv\_matnr as 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 as 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 retrieve matnr from the MARC table where matnr equals lv\_matnr\_sub.

7. Store the results in lt\_data and order by matnr.

8. Execute a SELECT SINGLE statement to retrieve lsobs from the MARD table into lv\_lsobs.

fetch\_vbuk

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

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

fetch\_orderby

1. Declare a string variable message.

2. Declare lv\_matnr and lv\_mtart as 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 retrieve matnr, mtart, and matkl from the mara table where matnr equals lv\_matnr\_sub10 and mtart equals lv\_mtart.

11. Store the results in lt\_table and order by matnr, mtart, and matkl.

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

fetch\_single

1. Declare lv\_matnr as type matnr.

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

3. Declare lv\_matnr\_sub as 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 retrieve matnr from the MARC table where matnr equals lv\_matnr\_sub.

6. Store the result in lv\_marc\_matnr.

get\_data (method)

1. Clear the variable gv\_vbrk.

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.

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: N/A

Output Destination: N/A

Description: The provided payload and explanations do not describe or reference any explicit output operations, such as report generation, file export, screen messages, or data display. All code and subroutines focus on data retrieval, processing, and internal table manipulation without any implemented output logic or destination.

# 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

[Flow diagram not available]

# 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 check for 'sy-subrc = 0' to determine if the SELECT was successful. However, no further logic, message display, or logging is implemented within this IF block; it serves as a placeholder for potential error handling but does not perform any explicit error handling or user notification.

2. In the subroutine 'fetch\_orderby', a message string is constructed by concatenating various variables when a specific condition is met (if 'lv\_matnr\_chk4' equals '1234'). This message is stored in a variable named 'message', but there is no indication in the explanation that this message is displayed to the user, logged, or otherwise output; it is only constructed and stored.

No other explicit error handling, exception processing, message display, or logging mechanisms are described or present in the provided explanations for the ABAP code.

# 15. Performance Considerations

1. The use of SELECT SINGLE statements in subroutines such as 'fetch\_and\_check\_plant', 'fetch\_dzaehk', 'fetch\_single', and 'fetch\_marc\_stawn' ensures that only one record is retrieved from the database, reducing data volume and improving performance when only a single entry is needed.

2. The SELECT statements in subroutines like 'fetch\_material', 'fetch\_delivery\_items', 'fetch\_konv', 'fetch\_bsak', 'fetch\_j1m0cust', and 'fetch\_jbbranch' utilize INTO CORRESPONDING FIELDS OF TABLE or INTO TABLE constructs, which directly populate internal tables, minimizing unnecessary data movement and enabling efficient bulk data retrieval.

3. The use of WHERE clauses in SELECT statements, such as filtering by material number in 'fetch\_material' and by substring in 'fetch\_single' and 'fetch\_marc\_mard', restricts the data retrieved to only relevant records, reducing the amount of data processed in memory.

4. The ORDER BY clause is used in several SELECT statements (e.g., 'fetch\_konv', 'fetch\_bsak', 'fetch\_j1m0cust', 'fetch\_jbbranch', 'fetch\_orderby', 'fetch\_material', 'fetch\_marc\_mard'), ensuring that data is sorted at the database level before being transferred to the application server, which is more efficient than sorting in ABAP.

5. Inline data declarations (e.g., @DATA) are used in SELECT statements to declare and fill internal tables or variables only when needed, which can help manage memory usage efficiently.

6. The use of FOR ALL ENTRIES in the 'fetch\_delivery\_items' subroutine allows batch selection of records from LIPS based on a set of material numbers and plant, reducing the number of database calls compared to individual SELECTs for each key.

7. The DESCRIBE TABLE statement in 'prepare\_final\_data' is used to determine the number of entries in an internal table, which can be useful for monitoring data volume and potentially controlling further processing.

8. The REFRESH statement is used to clear internal tables before reuse (e.g., 'ilips' in 'fetch\_delivery\_items', 'ifinal' in 'prepare\_final\_data'), ensuring that memory is managed and stale data does not accumulate.

9. Substring operations and conditional checks (e.g., in 'fetch\_orderby', 'fetch\_single', 'fetch\_marc\_mard') are performed before database access, allowing for more precise and limited data selection, which can reduce unnecessary data retrieval.

10. The use of SELECT COUNT(\*) in 'fetch\_vbuk' retrieves only the count of records from VBAK, avoiding the transfer of large data volumes when only the number of entries is required.

# 16. Security & Authorizations

|  |  |  |  |
| --- | --- | --- | --- |
| Object/Check Type | Name | Check Logic/Location | Description |
| [None] | [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 range | p\_werks = '1000', s\_matnr = ['MAT001', 'MAT002'] | Data retrieved and processed for plant 1000 and materials MAT001, MAT002; final output table populated | Pass | Standard positive scenario |
| Plant Not Provided | Ensure program handles missing plant input gracefully | p\_werks = '', s\_matnr = ['MAT001'] | No plant data fetched; dependent logic skipped or handled without error | Pass | Plant is optional; program should not dump |
| 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 | Material range is optional; program should not dump |
| Invalid Plant Code | Test program response to non-existent plant code | p\_werks = 'ZZZZ', s\_matnr = ['MAT001'] | No plant data found; error message or empty result as per logic | Pass | Error handling for invalid plant code |
| Invalid Material Number | Test program response to non-existent material number | p\_werks = '1000', s\_matnr = ['INVALID'] | No material data found; error message or empty result as per logic | Pass | Error handling for invalid material number |
| Both Inputs Empty | Ensure program handles both plant and material range missing | p\_werks = '', s\_matnr = [] | No data fetched; program completes without error | Pass | Program should not dump |
| Plant Provided, Material Range Invalid | Test program with valid plant and invalid material range | p\_werks = '1000', s\_matnr = ['INVALID'] | Plant data fetched; no material data; dependent logic handled gracefully | Pass | Partial data scenario |
| Material Range Provided, Plant Invalid | Test program with invalid plant and valid material range | p\_werks = 'ZZZZ', s\_matnr = ['MAT001'] | No plant data; material data fetched; dependent logic handled gracefully | Pass | Partial data scenario |
| Fetch Delivery Items with No Matching Data | Ensure delivery items fetch handles no matching records | p\_werks = '1000', s\_matnr = ['NONEXISTENT'] | No delivery items found; program completes without error | Pass | Negative data scenario |
| Fetch KONV with No Data | Test KONV fetch when prcd\_elements table is empty | prcd\_elements empty | ikonv internal table remains empty | Pass | Table empty scenario |
| Fetch VBRK with No Non-Draft Documents | Test VBRK fetch when all documents are drafts | VBRK table: all draft = 'X' | No vbeln fetched; variable remains initial | Pass | Edge case for SELECT SINGLE |
| Fetch VBRP with No Non-Draft Items | Test VBRP fetch when all items are drafts | VBRP table: all draft = 'X' | No items fetched; internal table remains empty | Pass | Edge case for SELECT |
| Fetch BSAK with No Data | Test BSAK fetch when ACDOCA table is empty | ACDOCA table empty | lt\_data internal table remains empty | Pass | Table empty scenario |
| Fetch J1M0CUST with No Data | Test customer fetch when KNA1 table is empty | KNA1 table empty | lt\_data internal table remains empty | Pass | Table empty scenario |
| Fetch MARC\_STAWN with No Data | Test MARC\_STAWN fetch when MARC table is empty | MARC table empty | ls\_marc initial; object methods called without MARC data | Pass | Table empty scenario |
| Fetch DZA EHK with No Data | Test DZA EHK fetch when prcd\_elements table is empty | prcd\_elements empty | lv\_dzaehk remains initial | Pass | Table empty scenario |
| Fetch JBBranch with No Data | Test JBBranch fetch when P\_BusinessPlace table is empty | P\_BusinessPlace table empty | lt\_data internal table remains empty | Pass | Table empty scenario |
| Fetch VBUK with No Data | Test VBUK fetch when VBAK table is empty | VBAK table empty | lv\_vbak\_cnt = 0 | Pass | Table empty scenario |
| Fetch MARC\_MARD with No Data | Test MARC\_MARD fetch when MARC and MARD tables are empty | MARC and MARD tables empty | lt\_data and lv\_lsobs remain initial | Pass | Table empty scenario |
| Fetch Orderby with No Matching Data | Test fetch\_orderby with non-matching material and type | lv\_matnr, lv\_mtart values not present in MARA | lt\_table remains empty; message logic handled gracefully | Pass | Negative data scenario |
| Prepare Final Data with Empty ilips | Test prepare\_final\_data when ilips is empty | ilips internal table empty | ifinal remains empty; no errors | Pass | Edge case for empty input table |
| Populate Salary with No ACDOCA Data | Test populate\_salary when acdoca-dmbtr is initial | acdoca-dmbtr initial or acdoca structure empty | lv\_salary = 0 or initial value | Pass | Edge case for missing data |
| Fetch Single with No Matching MARC | Test fetch\_single when substring does not match any MARC | lv\_matnr\_sub not present in MARC | lv\_marc\_matnr remains initial | Pass | Negative data scenario |
| Fetch MARC\_MARD with Substring Out of Bounds | Test fetch\_marc\_mard with lv\_matnr shorter than expected for substring | lv\_matnr = '12' | lv\_matnr\_sub = ''; SELECT returns no data; handled gracefully | Pass | Substring edge case |
| Fetch Orderby with Substring Out of Bounds | Test fetch\_orderby with lv\_matnr/lv\_mtart shorter than expected for substring | lv\_matnr = '12', lv\_mtart = 'A' | Substring variables = ''; SELECT returns no data; message logic handled gracefully | Pass | Substring edge case |
| Fetch Material with Large Range | Test performance and correctness with large material range | s\_matnr = ['MAT001' ... 'MAT999'] (large list) | All matching materials fetched into imara; performance acceptable | Pass | Performance scenario |
| Fetch Delivery Items with Large Data | Test performance and correctness with large ilips/imara | imara and i\_t001w-werks with many entries | All matching delivery items fetched and processed; performance acceptable | Pass | Performance scenario |
| Invalid Data Types on Selection Screen | Test input validation for wrong data types | p\_werks = '####', s\_matnr = ['!!!'] | Input validation error or no data fetched; program does not dump | Pass | Input validation scenario |
| SQL Error Handling | Simulate database error during SELECT | Database connection lost or table locked | Error handled gracefully; program does not dump | Pass | Error handling scenario |
| Authorization Error | Simulate lack of authorization for table access | User lacks authorization for MARA or LIPS | Authorization error message displayed; program does not dump | Pass | Security scenario |

# 18. Sign-Off

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

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