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 detail the design, structure, and logic of the program, which is developed as a modular ABAP report (type: PROG) utilizing multiple includes and subroutines. The scope covers the program’s data retrieval, processing, and output preparation functionalities, as well as the associated selection screen and supporting routines. 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 SAP ABAP program that efficiently retrieves, processes, and prepares data from various standard SAP tables such as MARA, LIPS, VBRK, VBRP, ACDOCA, and others. The solution must provide a user-friendly selection screen for inputting plant and material numbers, fetch relevant master and transactional data, perform necessary validations and transformations, and prepare the final dataset for reporting or further processing. The objective is to streamline data extraction and manipulation processes, ensuring accuracy, modularity, and reusability within the SAP ECC environment.

# 5. Solution Approach

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

2. Comprehensive Data Retrieval and Processing: The program leverages a series of PERFORM routines (subroutines) to encapsulate logic for fetching and processing data from key SAP tables (e.g., MARA, LIPS, VBRK, VBRP, ACDOCA). Each subroutine is responsible for a specific aspect of the data workflow, such as material selection, delivery item retrieval, condition record extraction, and final data preparation.

3. User-Driven Execution and Output Preparation: A selection screen enables users to specify input parameters (plant and material numbers), which drive the downstream data processing. The solution ensures that only relevant data is fetched and processed, culminating in the preparation of a final dataset suitable for reporting or further business use.

# 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 internal table definitions | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| Class | 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 |
| PERFORM Routine | fetch\_and\_check\_plant | Subroutine to check and fetch plant data from T001W | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_material | Subroutine to fetch material data from MARA | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_delivery\_items | Subroutine to fetch delivery item data from LIPS | 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 from VBRK | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_vbrp | Subroutine to fetch billing document items from VBRP | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_bsak | Subroutine to fetch financial data from ACDOCA | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_j1m0cust | Subroutine to fetch customer data from KNA1 | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_marc\_stawn | Subroutine to fetch MARC data and call commodity code services | 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 from P\_BusinessPlace | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |
| PERFORM Routine | fetch\_vbuk | Subroutine to count entries 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 from MARA 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 value 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, populated in subroutine 'fetch\_konv'. |
| 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'. |
| 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'. |
| t001w | t001w | Table work area declaration for database table T001W. | Used as a work area for plant data. |
| mara | mara | Table work area declaration for database table MARA. | Used as a work area for material master data. |
| lips | lips | Table work area declaration for database table LIPS. | Used as a work area for delivery item data. |
| 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, populated in 'fetch\_delivery\_items'. |
| 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'. |
| lv\_product | TYPE matnr | Variable of type MATNR for storing a product number. | Used for storing product/material numbers in various subroutines. |
| lv\_salary | TYPE dmbtr | Variable of type DMBTR for storing a salary amount. | Used for storing salary/amount values, e.g., in 'populate\_salary'. |
| 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. |
| vbrk | vbrk | SAP table: Billing Document Header | Accessed in subroutine 'fetc\_vbrk' to fetch billing document numbers. |
| vbrp | vbrp | SAP table: Billing Document Item | Accessed in subroutine 'fetch\_vbrp' to fetch billing document item data. |
| bsak | bsak | SAP table: Accounting: Secondary Index for Vendors (Cleared Items) | Accessed in subroutine 'fetch\_bsak' to fetch vendor open item data. |
| acdoca | acdoca | SAP table: Universal Journal Entry Line Items | Accessed in 'populate\_salary' to retrieve monetary values. |
| kna1 | kna1 | SAP table: General Data in Customer Master | Accessed in 'fetch\_j1m0cust' to fetch customer numbers. |
| prcd\_elements | prcd\_elements | SAP table: Pricing Document: Item Condition | Accessed in 'fetch\_konv' and 'fetch\_dzaehk' to fetch condition records and counters. |
| marc | marc | SAP table: Plant Data for Material | Accessed in 'fetch\_marc\_stawn', 'fetch\_marc\_mard', and 'fetch\_single' for material/plant data. |
| mard | mard | SAP table: Storage Location Data for Material | Accessed in 'fetch\_marc\_mard' to fetch storage location data. |
| lips | lips | SAP table: SD document: Delivery: Item data | Accessed in 'fetch\_delivery\_items' to fetch delivery item data. |
| t001w | t001w | SAP table: Plants/Branches | Accessed in 'fetch\_and\_check\_plant' to fetch plant data. |
| mara | mara | SAP table: General Material Data | Accessed in 'fetch\_material', 'fetch\_orderby', and as structure for internal tables. |
| vbak | vbak | SAP table: Sales Document: Header Data | Accessed in 'fetch\_vbuk' to count sales document headers. |
| P\_BusinessPlace | P\_BusinessPlace | SAP table: Business Place (custom or standard, as per context) | Accessed in 'fetch\_jbbranch' to fetch business place data. |
| /sapsll/cl\_mm\_cls\_service | ABAP Class Reference | Reference to commodity code classification service class. | Instantiated and used in 'fetch\_marc\_stawn' for commodity code operations. |
| vfprc\_cond\_count | Data Type | Data type for condition counter. | Used in 'fetch\_dzaehk' to store condition counter value. |
| gv\_vbrk | TYPE vbrk | Variable of type VBRK for storing billing document header data. | Used in class 'lcl\_data' and method 'get\_data'. |
| lt\_data | Internal Table (various types) | Inline-declared internal table for storing query results. | Used in multiple subroutines for temporary data storage (e.g., 'fetch\_bsak', 'fetch\_j1m0cust'). |
| lv\_vbeln | TYPE vbeln (inline @DATA) | Variable for storing billing document number. | Used in 'fetc\_vbrk' to store selected billing document number. |
| lt\_vbrk | Internal Table (inline @DATA) | Internal table for storing billing document items. | Used in 'fetch\_vbrp' to store selected billing document items. |
| ls\_marc | Structure (inline @DATA) | Structure for storing selected MARC fields. | Used in 'fetch\_marc\_stawn' to store selected MARC data. |
| lo\_cls\_service | TYPE REF TO /sapsll/cl\_mm\_cls\_service | Reference variable for commodity code classification service. | Used in 'fetch\_marc\_stawn' for method calls. |
| lo\_cls\_service\_det | TYPE REF TO /sapsll/cl\_mm\_cls\_service | Reference variable for commodity code classification service (details). | Used in 'fetch\_marc\_stawn' for method calls. |
| lv\_dzaehk | TYPE vfprc\_cond\_count | Variable for storing condition counter. | Used in 'fetch\_dzaehk' to store selected condition counter. |
| lv\_matnr | TYPE matnr | Variable for storing material number. | Used in multiple subroutines for material number processing. |
| lv\_matnr\_sub | TYPE char4/char3 | Variable for storing substring of material number. | Used in 'fetch\_marc\_mard', 'fetch\_single', and 'fetch\_orderby' for substring operations. |
| lv\_lsobs | TYPE mard-lsobs | Variable for storing storage location observation. | Used in 'fetch\_marc\_mard' to store selected MARD data. |
| message | TYPE string | Variable for storing concatenated message. | Used in 'fetch\_orderby' for message construction. |
| ty\_mara\_sel | Local Structure | Structure with fields matnr, mtart, matkl. | Used in 'fetch\_orderby' for SELECT result structure. |
| lt\_table | Internal Table of ty\_mara\_sel | Internal table for storing selected material data. | Used in 'fetch\_orderby' to store SELECT results. |
| lv\_matnr\_sub10 | TYPE c LENGTH 10 | Variable for storing substring of material number. | Used in 'fetch\_orderby' for substring operations. |
| lv\_matnr\_chk4 | TYPE c LENGTH 4 | Variable for storing substring of material number. | Used in 'fetch\_orderby' for substring operations and conditional logic. |
| lv\_matnr\_sub3 | TYPE c LENGTH 3 | Variable for storing substring of material number. | Used in 'fetch\_orderby' for substring operations. |
| lv\_mtart\_tail | TYPE c LENGTH 1 | Variable for storing substring of material type. | Used in 'fetch\_orderby' for substring operations. |
| lv\_lines | TYPE i | Variable for storing number of lines in internal table. | Used in 'prepare\_final\_data' to store count of entries in 'ifinal'. |
| lv\_marc\_matnr | TYPE matnr | Variable for storing selected material number from MARC. | Used in 'fetch\_single' to store SELECT result. |
| lcl\_data | Local Class | Local class for encapsulating data and methods. | Used for object-oriented encapsulation of data logic. |
| get\_data | Method | Method of class lcl\_data for data retrieval/processing. | Used to clear or process 'gv\_vbrk'. |

# 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, allowing the user to input a plant (p\_werks) and select material numbers (s\_matnr) for filtering data.

- At START-OF-SELECTION, the main processing block is triggered, executing a series of PERFORM statements that call specific subroutines in a defined sequence.

- 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 retrieves material data from the MARA table, selecting matnr, meins, and mtart fields into the internal table imara, filtered by the selected material numbers (s\_matnr).

- The fetch\_delivery\_items subroutine checks if both imara and i\_t001w-werks are not initial. If true, it clears the ilips table, performs a SELECT on LIPS for matching material numbers and plant, sorts the results, and transfers the data into ilips using MOVE-CORRESPONDING.

- The fetch\_konv subroutine retrieves knumv from the prcd\_elements table into the ikonv internal table, ordering by knumv, and checks if the SELECT was successful via sy-subrc.

- The fetc\_vbrk subroutine performs a SELECT SINGLE on VBRK to fetch vbeln where draft is empty, storing the result in lv\_vbeln.

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

- The fetch\_bsak subroutine selects multiple financial fields from ACDOCA into lt\_data, ordering by all selected fields.

- The fetch\_j1m0cust subroutine selects kunnr from KNA1 into lt\_data, ordered by kunnr.

- The fetch\_marc\_stawn subroutine selects a 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 these objects.

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

- The fetch\_jbbranch subroutine selects CompanyCode and BusinessPlace (as bukrs and branch) from P\_BusinessPlace into lt\_data, ordered by bukrs and branch.

- The fetch\_vbuk subroutine counts the number of entries in VBAK and stores the result in lv\_vbak\_cnt.

- The fetch\_marc\_mard subroutine extracts a substring from lv\_matnr, selects matnr from MARC where matnr matches the substring into lt\_data, and selects a single lsobs from MARD into lv\_lsobs.

- The fetch\_orderby subroutine declares several variables and substrings, selects matnr, mtart, and matkl from MARA into lt\_table where matnr and mtart match specific substrings, and if a substring equals '1234', concatenates a message string.

- The fetch\_single subroutine extracts a substring from lv\_matnr, selects a single matnr from MARC where matnr matches the substring into lv\_marc\_matnr.

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

- The populate\_salary subroutine assigns the value of acdoca-dmbtr to lv\_salary.

- Throughout the program, internal tables and structures are declared and manipulated to store and process data retrieved from the database.

- The flow is strictly sequential, with each subroutine performing its specific data retrieval or processing task, often conditioned on the presence of input parameters or the results of previous operations.

# 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 is provided, 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. Store 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 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 ilips internal table.

6. Execute a SELECT statement on the LIPS table to retrieve matnr, posnr, vbeln, and werks for all entries in imara where matnr equals imara-matnr and 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 over each entry in ilips.

3. For each entry:

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 to the internal table ifinal.

4. End the loop.

5. Use DESCRIBE TABLE to count the number of lines in ifinal and store in lv\_lines.

6. Refresh (clear) the ifinal internal table.

fetch\_konv

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

2. Store the results into the internal table ikonv, mapping corresponding fields.

3. Order the results by knumv.

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

5. 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 the field vbeln from the VBRK table where vbrk~draft equals space.

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

fetch\_vbrp

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

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

fetch\_bsak

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

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

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

fetch\_j1m0cust

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

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

3. Order the results by kunnr.

fetch\_marc\_stawn

1. Execute a SELECT SINGLE statement to retrieve the fields stawn and expme from the MARC table.

2. Store the result in an inline data object ls\_marc.

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

4. Create an instance of /sapsll/cl\_mm\_cls\_service and assign it to lo\_cls\_service.

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

6. Declare another reference variable lo\_cls\_service\_det of type /sapsll/cl\_mm\_cls\_service.

7. Create another instance of /sapsll/cl\_mm\_cls\_service and assign it to lo\_cls\_service\_det.

8. 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 the field condition\_counter from the prcd\_elements table.

3. Store the result in lv\_dzaehk.

fetch\_jbbranch

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

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

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 the lsobs field from the MARD table.

5. Assign to lv\_matnr\_sub a substring of lv\_matnr, starting at offset 3 and 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.

9. Store the result in lv\_lsobs.

fetch\_vbuk

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

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

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 a substring of lv\_matnr starting at position 4 with length 10.

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

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

9. Assign to lv\_mtart\_tail a 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 the strings 'Material', lv\_matnr, lv\_matnr\_sub3, 'Material', and lv\_mtart\_tail into the variable 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 a substring of lv\_matnr starting at offset 3 and 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. Execute the CLEAR statement on the variable gv\_vbrk, resetting its contents to the initial value.

lcl\_data (class 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.

lcl\_data (class implementation)

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

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

3. End the class implementation.

# 11. Output Details

[Error: Section Output Details not found after 3 retries.]

# 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 data into the internal table '@ikonv', there is an explicit check of the system field 'sy-subrc' to determine if the SELECT was successful (i.e., 'sy-subrc = 0'). However, there is no further logic, message display, or logging implemented within this check; it serves as a placeholder for potential error handling but does not perform any notification or logging action.

2. In the subroutine 'fetch\_orderby', there is conditional logic that, if a certain substring of the material number equals '1234', concatenates several strings into a variable 'message'. This is the only instance where a message variable is constructed, but there is no indication in the explanation that this message is displayed to the user, logged, or otherwise output; it is simply assigned.

# 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. Several subroutines, including 'fetch\_material', 'fetch\_konv', 'fetch\_j1m0cust', 'fetch\_bsak', and 'fetch\_jbbranch', utilize SELECT statements with INTO CORRESPONDING FIELDS OF TABLE or INTO TABLE clauses, which allow for direct population of internal tables and efficient bulk data retrieval.

3. The use of WHERE clauses in SELECT statements, such as in 'fetch\_material' (WHERE matnr IN @s\_matnr) and 'fetch\_orderby' (WHERE matnr = lv\_matnr\_sub10 AND mtart = lv\_mtart), restricts the data retrieved to only relevant records, minimizing unnecessary data transfer and processing.

4. ORDER BY clauses are used in SELECT statements within subroutines like 'fetch\_konv', 'fetch\_bsak', 'fetch\_j1m0cust', 'fetch\_orderby', and 'fetch\_marc\_mard' to ensure that data is sorted as required during retrieval, which can reduce the need for additional sorting in ABAP and improve downstream processing efficiency.

5. Inline data declarations (e.g., @DATA(lt\_data), @DATA(lv\_vbak\_cnt)) are used in several subroutines, which can help scope variables tightly and potentially reduce memory footprint during execution.

6. The use of FOR ALL ENTRIES in the 'fetch\_delivery\_items' subroutine enables set-based selection from the LIPS table based on the contents of another internal table, which can be more efficient than looping and selecting individually, provided the internal table is not empty.

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 controlling further processing.

8. The REFRESH statement is used to clear internal tables (e.g., 'ilips' and 'ifinal') before repopulation, ensuring that only current and relevant data is processed, which helps manage memory usage and avoids redundant data handling.

9. Substring operations and conditional checks (as seen in 'fetch\_orderby', 'fetch\_marc\_mard', and 'fetch\_single') are performed in ABAP before database access, which can help in constructing precise selection criteria and reduce the amount of data retrieved from the database.

# 16. Security & Authorizations

|  |  |  |  |
| --- | --- | --- | --- |
| Object/Check Type | Name | Check Logic/Location | Description |
| Authorization Object | [None] | [None] | [None] |
| Authority-Check | [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 = ['MAT001', 'MAT002'] | Data retrieved and processed for plant 1000 and materials MAT001, MAT002; final output tables populated | Pass | Standard positive scenario; all subroutines executed as expected. |
| Plant Not Provided | Ensure program handles missing plant input gracefully | p\_werks = '', s\_matnr = ['MAT001'] | Plant-related subroutines skipped or handled; no dump; program continues with material processing | Pass | Conditional logic in fetch\_and\_check\_plant covers this; no error expected. |
| Material Range Not Provided | Ensure program handles missing material input gracefully | p\_werks = '1000', s\_matnr = [] | Material-related subroutines skipped or handled; no dump; program continues with plant processing | Pass | SELECT-OPTIONS allows empty input; program should not fail. |
| Invalid Plant Code | Test program behavior with non-existent plant code | p\_werks = 'ZZZZ', s\_matnr = ['MAT001'] | No plant data found; dependent subroutines handle empty result; no dump | Pass | SELECT SINGLE returns no data; program logic should handle gracefully. |
| Invalid Material Numbers | Test program behavior with non-existent material numbers | p\_werks = '1000', s\_matnr = ['INVALID1', 'INVALID2'] | No material data found; dependent subroutines handle empty result; no dump | Pass | SELECT returns empty internal tables; program should not fail. |
| Both Plant and Material Inputs Empty | Ensure program handles both inputs missing | p\_werks = '', s\_matnr = [] | No data fetched; program completes without error | Pass | Edge case; program should not dump. |
| Plant Provided, Material Range with Single Value | Test program with plant and single material number | p\_werks = '1000', s\_matnr = ['MAT001'] | Data for plant 1000 and material MAT001 processed; final tables populated | Pass | Common scenario; all subroutines should execute. |
| Plant Provided, Material Range with Multiple Values | Test program with plant and multiple material numbers | p\_werks = '1000', s\_matnr = ['MAT001', 'MAT002', 'MAT003'] | Data for plant 1000 and all listed materials processed; final tables populated | Pass | Tests loop and range handling in SELECT-OPTIONS. |
| Plant Provided, Material Range with Invalid and Valid | Test program with plant and mix of valid/invalid material numbers | p\_werks = '1000', s\_matnr = ['MAT001', 'INVALID'] | Data for valid material(s) processed; invalid ignored; no dump | Pass | Ensures partial data is handled correctly. |
| SQL Error/Database Unavailable | Simulate database error during SELECT | p\_werks = '1000', s\_matnr = ['MAT001'] | Error handled gracefully; program does not dump; error message logged | N/A | Requires system manipulation; not covered in standard test but important for robustness. |
| Subroutine Logic Error (e.g., MOVE-CORRESPONDING fail) | Simulate logic error in subroutine | p\_werks = '1000', s\_matnr = ['MAT001'] | Program handles error gracefully; error message or skip record | N/A | Defensive programming should be in place; not directly testable via input. |
| Large Input Range | Test program with large range of material numbers | p\_werks = '1000', s\_matnr = ['MAT001'...'MAT999'] | All materials processed; performance acceptable; no memory issues | Pass | Performance and memory management test. |
| Duplicate Material Numbers in Input | Test program with duplicate material numbers in selection | p\_werks = '1000', s\_matnr = ['MAT001', 'MAT001', 'MAT002'] | Duplicates handled; no duplicate processing in output | Pass | SELECT-OPTIONS should handle duplicates; output should be unique. |
| Special Characters in Material Number | Test program with special characters in material number | p\_werks = '1000', s\_matnr = ['MAT@01', 'MAT#02'] | Program handles or rejects invalid characters; no dump | Pass | Input validation or error handling should be present. |
| Salary Population Logic | Test salary population subroutine with valid ACDOCA data | acdoca-dmbtr = 1000 | lv\_salary = 1000 | Pass | Ensures populate\_salary subroutine works as expected. |
| Salary Population with Missing ACDOCA Data | Test salary population subroutine with missing ACDOCA data | acdoca-dmbtr not available | lv\_salary remains initial or error handled | Pass | Subroutine should handle missing data gracefully. |
| Fetch Single Material with Short matnr | Test fetch\_single subroutine with matnr shorter than expected substring length | lv\_matnr = 'MA' | lv\_matnr\_sub = ''; SELECT SINGLE returns no data; no dump | Pass | Substring logic should not cause dump on short input. |
| Fetch Delivery Items with No Matching Data | Test fetch\_delivery\_items when no matching LIPS records | imara = valid, i\_t001w-werks = valid, but no matching LIPS records | ilips remains empty; no dump | Pass | Ensures empty result sets are handled. |
| Prepare Final Data with Empty ilips | Test prepare\_final\_data subroutine when ilips is empty | ilips = [] | ifinal remains empty; no dump | Pass | Loop should not execute; no errors. |
| Fetch KONV with No Data | Test fetch\_konv subroutine when prcd\_elements has no data | prcd\_elements empty | ikonv remains empty; no dump | Pass | SELECT returns empty; program should not fail. |

# 18. Sign-Off

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

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