Functional Specification Document

# Index

1. Document Information 1

2. Introduction 1

3. Business Requirement Overview 1

4. Business Process Flow 1

5. Functional Scope 1

6. Functional Solution Approach 1

7. Functional Requirements 1

8. Interfaces & Integration 1

9. Output 1

10. UI Requirement 1

11. Authorization & Security 1

12. Error Handling & Notifications 1

13. Assumptions & Dependencies 1

14. Test Scenario 1

15. Sign-Off 1

# 1. Document Information

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Document Title | Project Name | Client Name | Prepared By (Author, Department) | Document Version & Date |
| ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 Technical Specification | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |  | PWC AI Asset |  |

# 2. Introduction

This document provides the technical specification for the SAP ABAP program ZRCOPY\_SAMPLE\_ECC\_CODE\_V1. The purpose of this document is to outline the structure, logic, and components of the program, which is designed to perform a series of data retrieval and processing tasks within the SAP ECC environment. The scope includes the definition of selection screens, data declarations, subroutine logic, and the overall program flow as implemented in the provided ABAP code and includes. The intended audience for this document comprises SAP ABAP developers, technical consultants, and project stakeholders who require a detailed understanding of the program's technical design and implementation.

# 3. Business Requirement Overview

The business requirement addressed by the ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 program is to facilitate comprehensive data extraction and processing from various SAP standard tables such as MARA, LIPS, VBRK, VBRP, ACDOCA, KNA1, and others. The program aims to streamline the retrieval of material, plant, delivery, billing, financial, and customer data, as well as perform specific data manipulations and aggregations. The objective is to provide a modular and reusable solution that supports business processes requiring consolidated and validated data from multiple sources, thereby improving data accuracy, operational efficiency, and reporting capabilities within the SAP ECC system.

# 4. Business Process Flow

Start

-> User enters selection criteria (Plant and Material Numbers) on the selection screen

-> System retrieves plant data (fetch\_and\_check\_plant)

-> System fetches material master data (fetch\_material)

-> System fetches delivery item data (fetch\_delivery\_items)

-> System retrieves condition records (fetch\_konv)

-> System fetches billing document header (fetc\_vbrk)

-> System fetches billing document items (fetch\_vbrp)

-> System retrieves financial data (fetch\_bsak)

-> System fetches customer data (fetch\_j1m0cust)

-> System retrieves material storage and commodity code data (fetch\_marc\_stawn)

-> System fetches condition counter (fetch\_dzaehk)

-> System retrieves business place data (fetch\_jbbranch)

-> System counts sales document headers (fetch\_vbuk)

-> System fetches material and storage location data (fetch\_marc\_mard)

-> System performs ordered material data retrieval and message construction (fetch\_orderby)

-> System prepares final data set (prepare\_final\_data)

-> System populates salary data (populate\_salary)

-> End

# 5. Functional Scope

1. In-Scope items

1. The program 'ZRCOPY\_SAMPLE\_ECC\_CODE\_V1' is designed to perform a comprehensive set of data retrieval and processing tasks using ABAP subroutines (FORM routines) and object-oriented constructs.

2. The solution includes the following functional logic:

- Definition of a selection screen allowing users to input a plant (p\_werks) and select material numbers (s\_matnr) for data filtering.

- Data declarations for internal tables and structures to store and process information from SAP standard tables such as T001W (plant), MARA (material master), LIPS (delivery items), BSEG (accounting document segment), and others.

- Subroutines to fetch and validate plant data (fetch\_and\_check\_plant), retrieve material master data based on user input (fetch\_material), and fetch delivery item data filtered by material and plant (fetch\_delivery\_items).

- Retrieval of condition records from the PRCD\_ELEMENTS table (fetch\_konv), and billing document header and item data from VBRK and VBRP tables (fetc\_vbrk, fetch\_vbrp).

- Extraction of financial data from ACDOCA (fetch\_bsak), customer data from KNA1 (fetch\_j1m0cust), and business place data from P\_BusinessPlace (fetch\_jbbranch).

- Specialized subroutines for fetching commodity code and classification details from MARC and related services (fetch\_marc\_stawn), as well as material and storage location data from MARC and MARD (fetch\_marc\_mard).

- Logic to fetch and process number range or condition counter data from PRCD\_ELEMENTS (fetch\_dzaehk).

- Counting sales document headers in VBAK (fetch\_vbuk).

- Subroutines for advanced data processing, such as fetching and ordering material data with substring operations and conditional messaging (fetch\_orderby), and fetching single material numbers based on substrings (fetch\_single).

- Preparation of final output data by transferring and formatting data from internal tables (prepare\_final\_data).

- Population of salary-related data by extracting monetary values from ACDOCA (populate\_salary).

- Definition and minimal implementation of a local class (lcl\_data) with a public data variable and a method for data operations.

- All logic is modularized using includes and subroutines for maintainability and clarity.

2. Out-of-Scope items

1. The solution does not include any user interface beyond the selection screen; there are no ALV grids, custom screens, or interactive reports.

2. There is no logic for data updates, inserts, or deletions—only data retrieval and processing are in scope.

3. No error handling, exception management, or logging mechanisms are implemented in the provided code snippets.

4. There is no integration with external systems, web services, or RFC calls.

5. The implementation of business logic beyond data fetching, such as calculations, validations (other than basic checks like IS INITIAL), or workflow processing, is not included.

6. The code does not provide output formatting, printing, or exporting of results to files or other media.

7. No authorization checks or security-related logic are present.

8. The implementation of the method 'get\_data' in the class 'lcl\_data' is minimal and does not include any data retrieval or processing logic beyond clearing a variable.

9. The solution does not include any custom database tables or enhancements to standard SAP tables.

10. There is no documentation or code for scheduling, background processing, or job management.

# 6. Functional Solution Approach

The business requirement is addressed by designing an ABAP program that facilitates the retrieval, processing, and preparation of data from various SAP standard tables, based on user input provided via a selection screen. The solution begins with a selection screen where users specify a plant and a range of material numbers. Upon execution, the program sequentially performs a series of data extraction and processing steps, each encapsulated in dedicated subroutines for modularity and clarity.

The program first validates and fetches plant data if a plant code is provided. It then retrieves material master data for the selected materials, followed by fetching delivery item data filtered by the selected materials and plant. Condition records and billing document data are extracted from their respective tables. Additional subroutines fetch financial, customer, and branch data, as well as material-specific details such as commodity codes and storage location information.

Throughout the process, the program utilizes internal tables and structures to store intermediate results, ensuring data integrity and efficient processing. Substring operations and conditional logic are applied where necessary to derive specific data segments or to construct messages. The final step involves preparing a consolidated dataset for output or further processing, ensuring that only relevant and validated data is included. This approach ensures a comprehensive, flexible, and user-driven data extraction and processing workflow that meets the business requirements.

# 7. Functional Requirements

[Error: Section Functional Requirements not found in LLM output.]

# 8. Interfaces & Integration

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Interface ID | Source System | Target System | Data/Message Type | Frequency/Mode | Description |  |
|  | -------------- | -------------- | -------------- | ------------------- | ---------------- | ------------- |  |

# 9. Output

Based on the provided payload, the ABAP program 'ZRCOPY\_SAMPLE\_ECC\_CODE\_V1' is structured to perform a series of data retrieval and processing tasks through multiple subroutines. The outputs generated by this program are primarily internal tables and processed datasets, which are prepared for further use within the SAP system. The program fetches and processes data from various SAP tables such as MARA (material master), LIPS (delivery items), T001W (plant), PRCD\_ELEMENTS (pricing conditions), VBRK/VBRP (billing documents), ACDOCA (accounting documents), KNA1 (customers), VBAK (sales documents), MARD (storage locations), and custom or industry-specific tables/classes.

The main functional outputs include:

- Internal tables containing filtered and processed data for materials, deliveries, billing documents, accounting documents, customers, plants, and branches.

- A final internal table ('ifinal') that consolidates key fields (sales document number, item number, material number, plant) for reporting or further processing.

- Calculated or extracted values such as salary amounts, condition counters, and document counts.

- Messages or concatenated strings for logging or display purposes (e.g., in the 'fetch\_orderby' subroutine).

No explicit output format (such as Excel, PDF, or CSV) or destination (such as SAP AL11, email, or spool) is specified in the payload. The outputs are intended for internal use within the SAP system, likely for further reporting, display on the SAP GUI, or as input to subsequent processes.

The functional purpose of these outputs is to provide comprehensive, validated, and structured data extracts from core SAP modules (MM, SD, FI, etc.) to support business reporting, data analysis, and integration with other SAP or external processes.

# 10. UI Requirement

The selection screen consists of the following UI elements:

1. Field Name: p\_werks

- Type: Input (Single-value input field)

- Data Element: t001w-werks

- Description: Single-value input field for plant (WERKS) based on table T001W.

- Default Value: None specified

- Mandatory: Not specified

- Business Purpose/Validation: Used to specify a single plant. No additional validation or dependencies mentioned.

2. Field Name: s\_matnr

- Type: Input (Range input field)

- Data Element: mara-matnr

- Description: Range input for material number (MATNR) based on table MARA.

- Default Value: None specified

- Mandatory: Not specified

- Business Purpose/Validation: Used to specify a range of material numbers. No additional validation or dependencies mentioned.

No default values, mandatory flags, or inter-field dependencies are specified in the provided payload.

# 11. Authorization & Security

|  |  |  |  |
| --- | --- | --- | --- |
| Role/Profile | Authorization Object | Access Level | Description |
| [To Be Filled] | [To Be Filled] | [To Be Filled] | [To Be Filled] |

# 12. Error Handling & Notifications

# 13. Assumptions & Dependencies

# 14. Test Scenario

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Objective | Input Data | Expected Output | Actual Result/Status | Sign-off/Comments |
| TC01 | Validate plant selection and retrieval from T001W | p\_werks = '1000' | Plant data for WERKS = '1000' is fetched into i\_t001w |  |  |
| TC02 | Validate material range selection and retrieval from MARA | s\_matnr = ['MAT1', 'MAT2', 'MAT3'] | Materials MAT1, MAT2, MAT3 with fields matnr, meins, mtart are fetched into imara |  |  |
| TC03 | Fetch delivery items for selected materials and plant | imara populated, i\_t001w-werks = '1000' | ilips internal table populated with delivery items matching materials and plant |  |  |
| TC04 | Prepare final data from delivery items | ilips populated | ifinal internal table populated with vbeln, posnr, matnr, werks from ilips |  |  |
| TC05 | Fetch condition records from prcd\_elements | None (table data exists) | ikonv internal table populated with knumv values from prcd\_elements |  |  |
| TC06 | Fetch billing document header where draft is space | VBRK table with at least one record where draft = space | lv\_vbeln variable populated with vbeln from VBRK |  |  |
| TC07 | Fetch billing document items where draft is space | VBRP table with at least one record where draft = space | lt\_vbrk internal table populated with vbeln, posnr from VBRP |  |  |
| TC08 | Fetch financial data from ACDOCA table | ACDOCA table with data | lt\_data internal table populated with bukrs, gjahr, belnr, buzei, hkont, dmbtr, wrbtr, waers, budat |  |  |
| TC09 | Fetch customer numbers from KNA1 | KNA1 table with data | lt\_data internal table populated with kunnr, ordered by kunnr |  |  |
| TC10 | Fetch MARC stawn and call commodity code services | MARC table with data | ls\_marc populated with stawn, expme; /sapsll/cl\_mm\_cls\_service methods called |  |  |
| TC11 | Fetch condition counter from prcd\_elements | prcd\_elements table with data | lv\_dzaehk populated with condition\_counter |  |  |
| TC12 | Fetch business place data from P\_BusinessPlace | P\_BusinessPlace table with data | lt\_data internal table populated with bukrs, branch, ordered by both |  |  |
| TC13 | Count number of sales document headers in VBAK | VBAK table with data | lv\_vbak\_cnt populated with count of VBAK entries |  |  |
| TC14 | Fetch MARC material numbers and MARD lsobs | MARC and MARD tables with data | lt\_data populated with matnr from MARC; lv\_lsobs populated with lsobs from MARD |  |  |
| TC15 | Fetch material data with substring and ordering logic | lv\_matnr and lv\_mtart set; MARA table with matching data | lt\_table populated with matnr, mtart, matkl; message variable set if condition met |  |  |
| TC16 | Fetch single material number from MARC using substring | lv\_matnr set; MARC table with matching matnr | lv\_marc\_matnr populated with matnr from MARC |  |  |
| TC17 | Populate salary from ACDOCA | acdoca-dmbtr field populated | lv\_salary variable set to value of acdoca-dmbtr |  |  |
| TC18 | Class lcl\_data: get\_data method clears gv\_vbrk | gv\_vbrk set to non-initial value | After get\_data call, gv\_vbrk is initial |  |  |

# 15. Sign-Off

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

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