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 |
| Functional Specification for ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 | ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 |  | PWC AI Asset | 1.0 / 2024-06-10 |

# 2. Introduction

This document serves as the functional specification for the SAP ABAP program ZRCOPY\_SAMPLE\_ECC\_CODE\_V1. The purpose of this document is to outline the functional requirements, scope, and intended audience for the development and implementation of the program. The scope includes the retrieval, processing, and preparation of data from various SAP standard tables such as MARA, LIPS, T001W, VBRK, VBRP, ACDOCA, and others, as well as the execution of business logic encapsulated in modular subroutines and classes. The intended audience for this document includes SAP functional consultants, ABAP developers, business analysts, and project stakeholders involved in the design, development, and validation of the solution.

# 3. Business Requirement Overview

The business requires a comprehensive SAP ABAP solution to efficiently retrieve, process, and prepare data from multiple standard SAP tables, including material master, delivery, billing, accounting, and customer information. The current process involves manual or fragmented data extraction and manipulation, leading to inefficiencies and potential data inconsistencies. The objective of the proposed solution is to automate and streamline these operations through a modular ABAP program (ZRCOPY\_SAMPLE\_ECC\_CODE\_V1) that leverages selection screens, internal tables, subroutines, and object-oriented constructs. This will enable accurate, timely, and consistent data processing to support downstream business processes and reporting requirements.

# 4. Business Process Flow

Start

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

-> System validates and fetches plant data (fetch\_and\_check\_plant)

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

-> System fetches delivery items based on material and plant (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 accounting data (fetch\_bsak)

-> System fetches customer data (fetch\_j1m0cust)

-> System fetches MARC commodity code and details (fetch\_marc\_stawn)

-> System fetches condition counter (fetch\_dzaehk)

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

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

-> System fetches MARC and MARD 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 solution includes the setup and declaration of internal tables and variables for processing SAP data, such as delivery items, material master data, and monetary values. Specifically, internal tables are created for structures like LIPS (delivery items), MARA (material master), and KONV (condition records), as well as variables for material numbers and amounts.

2. The program provides a selection screen for user input, allowing users to specify a plant (P\_WERKS) and a range of material numbers (S\_MATNR) for filtering data.

3. The main program logic is modularized using subroutines (PERFORM statements), each responsible for a specific data retrieval or processing task. These subroutines include:

- Fetching and validating plant data from T001W.

- Fetching material data from MARA based on user selection.

- Fetching delivery item data from LIPS, filtered by material and plant.

- Fetching condition records from PRCD\_ELEMENTS (KONV).

- Fetching billing document headers (VBRK) and items (VBRP), ensuring only non-draft documents are considered.

- Fetching financial data from ACDOCA, including company code, fiscal year, document number, line item, GL account, amounts, currency, and posting date.

- Fetching customer numbers from KNA1.

- Fetching commodity code and details from MARC and using /SAPSLL/CL\_MM\_CLS\_SERVICE class methods.

- Fetching condition counters from PRCD\_ELEMENTS.

- Fetching business place data from P\_BusinessPlace.

- Counting entries in VBAK (sales document headers).

- Fetching material and storage location data from MARC and MARD, including substring operations on material numbers.

- Fetching and processing material data with ordering and substring logic from MARA.

- Fetching a single material number from MARC based on a substring of another material number.

- Preparing final data by transferring and appending relevant fields from delivery items to the final output table, counting entries, and refreshing the table.

- Populating salary data by extracting monetary amounts from ACDOCA.

4. The solution defines a local class (LCL\_DATA) with a public data variable for billing document headers and a method for data retrieval, though the method implementation is minimal.

5. The program structure is organized using INCLUDES for modularity, separating variable declarations, selection screen definitions, and subroutine implementations.

2. Out-of-Scope items

1. The solution does not include any user interface enhancements beyond the basic selection screen (no ALV grids, custom dialogs, or advanced UI features).

2. There is no implementation of error handling, logging, or exception management within the subroutines or main program logic.

3. The program does not provide any output formatting, reporting, or data export functionality (such as generating files or displaying results in a specific layout).

4. There is no integration with external systems, web services, or non-SAP data sources.

5. The solution does not include any business logic for data validation, authorization checks, or workflow integration.

6. No enhancements, BADIs, user exits, or modifications to standard SAP objects are included.

7. The implementation does not cover performance optimization, parallel processing, or background job scheduling.

8. There is no coverage of data update, insert, or delete operations—only data retrieval and internal processing are in scope.

9. The solution does not include documentation, training materials, or end-user support processes.

10. There is no coverage of transport management, version control, or deployment procedures.

# 6. Functional Solution Approach

The business requirement will be addressed by designing an ABAP program that orchestrates a series of data retrieval and processing steps using modular subroutines and object-oriented constructs. The program begins by presenting a selection screen to the user, allowing input of plant and material numbers, which are then used as filters for subsequent data extraction. Upon execution, the program sequentially performs the following functional steps:

First, it validates and fetches plant data based on user input, ensuring that only relevant plant records are considered. Material master data is then retrieved for the selected materials, followed by the extraction of delivery item data that matches both the selected materials and plant. The program continues by fetching condition records, billing document headers and items, vendor open items, customer data, and additional master data such as commodity codes, branches, and storage locations. Each subroutine is responsible for a specific data domain, ensuring clear separation of concerns and maintainability.

Throughout the process, the program utilizes internal tables and structures to store intermediate results, leveraging both procedural and object-oriented ABAP features. Data is sorted, filtered, and transformed as needed, including substring operations and conditional logic to derive specific values. The final step involves preparing a consolidated dataset that combines the relevant fields from the various sources, ready for output or further processing. This approach ensures that the business requirement is met through a robust, modular, and reusable ABAP solution that efficiently integrates data from multiple SAP standard tables.

# 7. Functional Requirements

[Error: Section Functional Requirements not found after 3 retries.]

# 8. Interfaces & Integration

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Interface ID | Source System | Target System | Data/Message Type | Frequency/Mode | Description |
| [To Be Filled] | [To Be Filled] | [To Be Filled] | [To Be Filled] | [To Be Filled] | [To Be Filled] |

# 9. Output

The program 'ZRCOPY\_SAMPLE\_ECC\_CODE\_V1' is designed to perform a series of data retrieval and processing tasks across various SAP tables, culminating in the preparation of a final dataset. The functional outputs generated by this program include:

- Extraction of plant, material, delivery item, pricing condition, billing document, vendor, customer, commodity code, branch, and sales document status data from standard SAP tables such as T001W, MARA, LIPS, KONV, VBRK, VBRP, BSAK, KNA1, MARC, MARD, PRCD\_ELEMENTS, and custom tables/views.

- Preparation of a final internal table ('ifinal') that consolidates key fields such as sales document number, item number, material number, and plant for further processing or reporting.

- The program is structured to process and organize data for reporting or downstream consumption, but the specific output format (such as Excel, PDF, CSV) and destination (such as SAP AL11, email, or spool) are not explicitly specified in the provided payload.

- The functional purpose of the output is to provide a comprehensive, processed dataset that can be used for reporting, analysis, or integration with other SAP or external systems, based on the extracted and validated data from multiple SAP modules (MM, SD, FI, etc.).

# 10. UI Requirement

The selection screen includes the following UI elements:

- p\_werks: This is a single-value input field for plant (WERKS) based on table T001W. The user can enter a single plant code. The field type is Input. No default value or mandatory status is specified in the payload. The business purpose is to allow the user to filter data by a specific plant.

- s\_matnr: This is a range input for material number (MATNR) based on table MARA. The user can enter a single value, multiple values, or a range of material numbers. The field type is Input (Range/Select-Options). No default value or mandatory status is specified in the payload. The business purpose is to enable filtering by one or more material numbers.

No additional validations, dependencies, or interactivity between fields are specified in the 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

1. [To Be Filled]

# 13. Assumptions & Dependencies

1. The program relies on the existence and structure of standard SAP tables such as T001W, MARA, LIPS, BSEG, KONV, VBRK, VBRP, ACDOCA, KNA1, PRCD\_ELEMENTS, MARC, MARD, VBAK, and P\_BusinessPlace.

2. Selection screen parameters and select-options are assumed to be provided by the user for plant (WERKS) and material number (MATNR).

3. The includes (e.g., ZRCOPY\_SAMPLE\_ECC\_CODE\_TOP\_V1, ZRCOPY\_SAMPLE\_ECC\_CODE\_F01\_V1) are assumed to be present and contain the necessary declarations and subroutine implementations.

4. The referenced classes (e.g., /sapsll/cl\_mm\_cls\_service) and their methods are assumed to be available in the system.

5. The program assumes that the user has the necessary authorizations to access the relevant SAP tables and execute the required operations.

# 14. Test Scenario

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Objective | Input Data | Expected Output | Actual Result/Status | Sign-off/Comments |
| TC01 | Validate selection screen accepts plant and material number inputs | p\_werks = '1000', s\_matnr = ['MAT1', 'MAT2'] | Selection screen displays single-value input for plant and range input for material numbers; user can enter values |  |  |
| TC02 | Ensure plant data is fetched and validated when plant input is provided | p\_werks = '1000' | i\_t001w is populated with plant data from T001W for '1000' |  |  |
| TC03 | Fetch material data based on selected material numbers | s\_matnr = ['MAT1', 'MAT2'] | imara internal table is populated with MARA records for 'MAT1' and 'MAT2' |  |  |
| TC04 | Fetch delivery items for selected materials and plant | imara populated, i\_t001w-werks = '1000' | ilips internal table is filled with LIPS records matching materials and plant |  |  |
| TC05 | Prepare final data from delivery items | ilips populated | ifinal internal table is filled with vbeln, posnr, matnr, werks from ilips |  |  |
| TC06 | Fetch condition records from prcd\_elements |  | ikonv internal table is populated with knumv from prcd\_elements, sorted by knumv |  |  |
| TC07 | Fetch billing document header where draft is space |  | lv\_vbeln is set to vbeln from VBRK where draft = space |  |  |
| TC08 | Fetch billing document items where draft is space |  | lt\_vbrk internal table is filled with vbeln, posnr from VBRP where draft = space |  |  |
| TC09 | Fetch financial data from ACDOCA |  | lt\_data internal table is filled with bukrs, gjahr, belnr, buzei, hkont, dmbtr, wrbtr, waers, budat from ACDOCA |  |  |
| TC10 | Fetch customer numbers from KNA1 |  | lt\_data internal table is filled with kunnr from KNA1, ordered by kunnr |  |  |
| TC11 | Fetch MARC stawn and call commodity code services |  | ls\_marc is filled with stawn, expme from MARC; /sapsll/cl\_mm\_cls\_service methods are called |  |  |
| TC12 | Fetch condition counter from prcd\_elements |  | lv\_dzaehk is set to condition\_counter from prcd\_elements |  |  |
| TC13 | Fetch business place data from P\_BusinessPlace |  | lt\_data internal table is filled with bukrs, branch from P\_BusinessPlace, ordered by both |  |  |
| TC14 | Fetch material numbers from MARC and lsobs from MARD | lv\_matnr = 'MAT123456' | lt\_data is filled with matnr from MARC matching substring, lv\_lsobs is set from MARD |  |  |
| TC15 | Count entries in VBAK |  | lv\_vbak\_cnt is set to count of entries in VBAK |  |  |
| TC16 | Fetch material data with substring and ordering logic | lv\_matnr, lv\_mtart set | lt\_table is filled with matnr, mtart, matkl from MARA matching substring and type, message is constructed if condition met |  |  |
| TC17 | Fetch single material number from MARC using substring | lv\_matnr = 'MAT123456' | lv\_marc\_matnr is set to matnr from MARC matching substring |  |  |
| TC18 | Populate salary from ACDOCA | acdoca-dmbtr = 1000 | lv\_salary is set to 1000 |  |  |

# 15. Sign-Off

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

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