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 Project |  | 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 comprehensive data retrieval and processing tasks within the SAP ECC environment. The scope includes detailed explanations of the program’s modular subroutines, data structures, and selection screen, as well as the integration of various SAP standard tables such as MARA, LIPS, VBRK, and others. The intended audience for this document includes SAP ABAP developers, technical consultants, and project stakeholders who require a clear understanding of the program’s technical design and implementation approach.

# 3. Business Requirement Overview

The business requirement addressed by the ZRCOPY\_SAMPLE\_ECC\_CODE\_V1 program is to automate and streamline the extraction, processing, and preparation of key business data from various SAP modules. The program is designed to fetch and process data related to materials, deliveries, billing documents, financial postings, customer and plant information, and other relevant business entities. The objective is to provide a modular, reusable, and efficient solution that supports business reporting, validation, and integration needs by leveraging SAP standard tables and structured ABAP subroutines. This approach aims to reduce manual effort, improve data accuracy, and enable timely access to critical business information for decision-making.

# 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 based on selection (fetch\_material)

-> System fetches delivery item data for selected materials 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 fetches financial posting data (fetch\_bsak)

-> System retrieves customer master data (fetch\_j1m0cust)

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

-> System retrieves condition counter (fetch\_dzaehk)

-> System fetches business place/branch 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 output data (prepare\_final\_data)

-> System populates salary-related 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 (PERFORM statements). The following functionalities are included within the scope:

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

- Data declarations and setup of internal tables for storing and processing data related to materials, deliveries, plants, and conditions.

- Fetching and validating plant data from T001W based on user input.

- Retrieving material master data from MARA for selected material numbers.

- Fetching delivery item data from LIPS based on selected materials and plant, and organizing this data for further processing.

- Retrieving condition records from the PRCD\_ELEMENTS table and storing them in an internal table.

- Fetching billing document header (VBRK) and item (VBRP) data, specifically for non-draft documents.

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

- Retrieving customer numbers from KNA1 and business place data from P\_BusinessPlace.

- Fetching material storage and commodity code data from MARC and MARD, including the use of object-oriented service classes for commodity code classification.

- Fetching and processing number range or condition counter data from PRCD\_ELEMENTS.

- Counting the number of sales document headers in VBAK.

- Performing substring operations on material numbers and types for dynamic data selection and message construction.

- Preparing a final dataset by processing delivery item data and organizing it into a structured internal table.

- Populating salary-related data by extracting monetary values from ACDOCA.

- Modularization of logic using includes and subroutines for maintainability and reusability.

- Use of both procedural (FORM routines) and object-oriented (local class lcl\_data) ABAP constructs.

2. Out-of-Scope items

1. The following functionalities are explicitly or implicitly excluded from the scope:

- Any user interface beyond the selection screen (no ALV grid, custom screens, or interactive reports).

- Data manipulation or updates to the database tables; all operations are read-only (SELECT statements only, no INSERT/UPDATE/DELETE).

- Error handling, logging, or exception management beyond basic SY-SUBRC checks.

- Authorization checks or user-specific data filtering.

- Integration with external systems or non-SAP data sources.

- Advanced business logic such as pricing calculations, workflow triggers, or document posting.

- Output formatting, printing, or exporting of results (no explicit output routines are defined).

- Performance optimization techniques such as parallel processing, buffering, or advanced SQL tuning.

- Detailed implementation of the methods in the local class lcl\_data beyond basic structure (no business logic in get\_data).

- Any HR-specific logic beyond the simple extraction of salary amounts from ACDOCA.

- Custom enhancements, BADIs, or user exits.

- Handling of special cases such as archiving, deletion flags, or data inconsistencies.

# 6. Functional Solution Approach

The business requirement is addressed by structuring the ABAP program 'ZRCOPY\_SAMPLE\_ECC\_CODE\_V1' to guide users through a series of data retrieval and processing steps, driven by user input on a selection screen. The solution begins with a selection screen where users specify a plant and a range of material numbers, ensuring that only relevant data is processed. The program then sequentially executes a set of subroutines, each responsible for fetching and processing data from key SAP tables such as T001W (plant), MARA (material master), LIPS (delivery items), PRCD\_ELEMENTS (pricing conditions), VBRK/VBRP (billing documents), ACDOCA (accounting), KNA1 (customers), and others.

Each subroutine encapsulates a specific business function, such as validating plant input, retrieving material and delivery item data, fetching pricing and billing information, and preparing final output datasets. The use of internal tables and structured data movement ensures that data integrity is maintained throughout the process. Object-oriented techniques are also employed, as seen in the definition and use of local classes for encapsulating data and methods.

The program concludes by preparing the final dataset for output or further processing, ensuring that all user selections and business rules are respected. This modular and sequential approach allows for clear traceability, ease of maintenance, and the ability to extend or modify individual business functions as requirements evolve.

# 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

[Error: Section Output not found in LLM output.]

# 10. UI Requirement

Selection Screen Fields:

1. Field Name: p\_werks

- Type: Input (Single-value)

- 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 filter data by plant. No additional validation or dependencies mentioned.

2. Field Name: s\_matnr

- Type: Input (Range)

- 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 filter data by material number. No additional validation or dependencies mentioned.

No other UI elements, default values, mandatory flags, or inter-field dependencies 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

# 13. Assumptions & Dependencies

# 14. Test Scenario

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Objective | Input Data | Expected Output | Actual Result/Status | Sign-off/Comments |
| TC01 | Validate plant input and fetch plant data | p\_werks = '1000' | Plant data for WERKS '1000' is fetched from T001W and stored in i\_t001w |  |  |
| TC02 | Fetch material master data for a range of material numbers | s\_matnr = ['MAT001', 'MAT002'] | Material data for MAT001 and MAT002 is fetched from MARA into imara |  |  |
| TC03 | Fetch delivery items for selected materials and plant | imara contains MAT001, MAT002; i\_t001w-werks = '1000' | Delivery items from LIPS for MAT001, MAT002 and plant '1000' are fetched into ilips |  |  |
| TC04 | Prepare final data from delivery items | ilips populated with delivery items | ifinal internal table is populated with vbeln, posnr, matnr, werks from ilips |  |  |
| TC05 | Fetch condition records from prcd\_elements | No specific input | ikonv internal table is populated with knumv from prcd\_elements, sorted by knumv |  |  |
| TC06 | Fetch billing document header where draft is empty | No specific input | lv\_vbeln is populated with vbeln from VBRK where draft = space |  |  |
| TC07 | Fetch billing document items where draft is empty | No specific input | lt\_vbrk internal table is populated with vbeln, posnr from VBRP where draft = space |  |  |
| TC08 | Fetch accounting document data from ACDOCA | No specific input | lt\_data internal table is populated with bukrs, gjahr, belnr, buzei, hkont, dmbtr, wrbtr, waers, budat from ACDOCA, ordered |  |  |
| TC09 | Fetch customer numbers from KNA1 | No specific input | lt\_data internal table is populated with kunnr from KNA1, ordered by kunnr |  |  |
| TC10 | Fetch MARC stawn and call commodity code services | No specific input | ls\_marc is populated with stawn, expme from MARC; /sapsll/cl\_mm\_cls\_service methods are called |  |  |
| TC11 | Fetch condition counter from prcd\_elements | No specific input | lv\_dzaehk is populated with condition\_counter from prcd\_elements |  |  |
| TC12 | Fetch business place data from P\_BusinessPlace | No specific input | lt\_data internal table is populated with bukrs, branch from P\_BusinessPlace, ordered by both fields |  |  |
| TC13 | Count number of sales document headers in VBAK | No specific input | lv\_vbak\_cnt is populated with the count of entries in VBAK |  |  |
| TC14 | Fetch MARC material numbers and MARD lsobs | lv\_matnr = 'MAT1234567' | lt\_data is populated with matnr from MARC where matnr = substring(lv\_matnr,3,4); lv\_lsobs is populated with lsobs from MARD |  |  |
| TC15 | Fetch material data with ordering and substring checks | lv\_matnr, lv\_mtart set; lv\_matnr\_chk4 = '1234' | lt\_table is populated with matnr, mtart, matkl from MARA where matnr = lv\_matnr\_sub10 and mtart = lv\_mtart; message is concatenated |  |  |
| TC16 | Fetch single MARC material number by substring | lv\_matnr = 'MAT1234567' | lv\_marc\_matnr is populated with matnr from MARC where matnr = substring(lv\_matnr,3,3) |  |  |
| TC17 | 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.