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 | [To Be Filled] | [To Be Filled] | PWC AI Asset | [To Be Filled] |

# 2. Introduction

This document serves as the functional specification for the ABAP program "ZRCOPY\_SAMPLE\_ECC\_CODE\_V1" (Type: PROG). The purpose of this document is to define the scope, objectives, and requirements for the development and implementation of the program, which is designed to perform comprehensive data retrieval and processing tasks across various SAP standard and custom tables. The scope includes the setup of selection screens, data declarations, internal table structures, and a series of subroutines for fetching and processing data related to materials, deliveries, billing, financials, and other business objects. 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 an ABAP program that facilitates the extraction, validation, and processing of key business data from multiple SAP modules, including Materials Management (MM), Sales and Distribution (SD), and Financial Accounting (FI). The current process involves manual or fragmented data retrieval from various tables such as MARA, LIPS, VBRK, VBRP, ACDOCA, and others, leading to inefficiencies, potential data inconsistencies, and increased processing time.

The objective of the proposed solution is to automate and streamline the data retrieval and processing workflow by providing a single program with a user-friendly selection screen. The program will enable users to input plant and material selection criteria, fetch relevant data from multiple SAP tables, perform necessary validations and transformations, and prepare the final dataset for reporting or further processing. This approach aims to improve data accuracy, reduce manual effort, and enhance overall process efficiency.

# 4. Business Process Flow

User Inputs Selection Criteria (Plant, Material) -> Fetch and Validate Plant Data -> Fetch Material Master Data -> Fetch Delivery Items -> Fetch Pricing/Condition Records -> Fetch Billing Header and Item Data -> Fetch Financial and Customer Data -> Fetch Commodity Code and Branch Data -> Fetch Additional Material and Sales Data -> Prepare Final Output Dataset -> End

# 5. Functional Scope

1. In-Scope Items:

1.1. Data Declarations and Internal Table Setup:

- Declaration and initialization of internal tables for processing delivery (LIPS), material master (MARA), pricing conditions (KONV), and plant data (T001W).

- Use of structures and header lines for internal tables to facilitate data manipulation.

1.2. Selection Screen Functionality:

- User input for plant (P\_WERKS) and material number range (S\_MATNR) via selection screen.

- Grouping of selection screen fields for better user experience.

1.3. Data Retrieval and Processing Subroutines:

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

- Retrieval of material master data (MARA) filtered by user-selected material numbers.

- Extraction of delivery item data (LIPS) based on selected materials and plant, with sorting and transfer to internal tables.

- Preparation of final data set by processing delivery items and populating a result internal table (IFINAL).

- Fetching of pricing condition records (KONV/PRCD\_ELEMENTS) and sorting them.

- Retrieval of billing document header (VBRK) and item (VBRP) data, including filtering for non-draft documents.

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

- Fetching of customer master data (KNA1) and business place data (P\_BusinessPlace), with sorting.

- Retrieval of material storage location and commodity code data from MARC and MARD, including substring operations on material numbers.

- Use of object-oriented ABAP for commodity code classification via /SAPSLL/CL\_MM\_CLS\_SERVICE.

- Counting of sales document headers in VBAK.

- Data ordering and message construction based on material number and type substrings.

- Population of salary data from ACDOCA into local variables.

1.4. Modular Program Structure:

- Use of includes for modularization (TOP, F01, SEL).

- Encapsulation of logic in subroutines (FORM/ENDFORM) and local classes (LCL\_DATA).

- Use of PERFORM statements to orchestrate the main processing flow.

1.5. Object-Oriented Enhancements:

- Definition and minimal implementation of a local class (LCL\_DATA) with a public data attribute and method.

- Placeholder for future object-oriented logic (method GET\_DATA).

1.6. Data Preparation and Output:

- Preparation of final output data set by aggregating and transforming data from various sources.

- Counting and clearing of internal tables as part of data lifecycle management.

2. Out-of-Scope Items:

2.1. User Interface Enhancements:

- No advanced ALV grid display, custom screen layouts, or interactive UI elements beyond the basic selection screen.

2.2. Error Handling and Logging:

- No explicit error handling, exception management, or logging mechanisms for failed database operations or invalid user input.

2.3. Authorization and Security:

- No checks for user authorizations, data access restrictions, or security validations.

2.4. Data Update or Posting:

- No logic for updating, inserting, or deleting records in SAP tables; all operations are read-only.

2.5. Integration with External Systems:

- No interfaces or data exchange with external systems, web services, or non-SAP platforms.

2.6. Performance Optimization:

- No explicit performance tuning, parallel processing, or buffering strategies.

2.7. Advanced Business Logic:

- No complex business rules, calculations, or workflow integration beyond basic data retrieval and transfer.

2.8. Output Formatting and Reporting:

- No formatted report output, printing, or export to external formats (e.g., Excel, PDF).

2.9. Unit Testing and Test Data Management:

- No test harness, unit test classes, or test data generation logic.

2.10. Custom Table Maintenance:

- No logic for maintaining or customizing Z-tables or custom configuration tables.

# 6. Functional Solution Approach

The functional solution for the program "ZRCOPY\_SAMPLE\_ECC\_CODE\_V1" is designed to address a comprehensive data extraction and processing requirement within the SAP ECC environment. The approach is modular, leveraging both procedural and object-oriented ABAP constructs, and is structured as follows:

1. \*\*User Input and Selection Criteria\*\*

- The program presents a selection screen to the user, allowing input of a plant (P\_WERKS) and a range of material numbers (S\_MATNR). This enables flexible and targeted data retrieval based on user-defined criteria.

2. \*\*Data Structure Preparation\*\*

- Internal tables and structures are declared to mirror key SAP tables (e.g., MARA for materials, LIPS for deliveries, T001W for plants, KONV for pricing conditions). These structures facilitate efficient data storage and manipulation throughout the program.

3. \*\*Plant Validation and Retrieval\*\*

- The program first checks if a plant has been provided. If so, it validates the plant code and fetches the corresponding plant data from T001W, ensuring that subsequent processing is restricted to valid and relevant plant information.

4. \*\*Material Data Extraction\*\*

- Material master data is retrieved from MARA based on the selected material numbers. This ensures that only materials relevant to the user's selection are processed further.

5. \*\*Delivery Item Data Extraction\*\*

- Delivery item data is fetched from LIPS, filtered by the selected materials and plant. This step links material and plant data to actual delivery transactions, forming the basis for downstream processing.

6. \*\*Pricing Condition Retrieval\*\*

- Pricing condition records are extracted from PRCD\_ELEMENTS (serving the role of KONV), ordered and stored for further use. This enables the program to associate pricing information with delivery and material data.

7. \*\*Billing Document Processing\*\*

- The program retrieves billing document headers (VBRK) and items (VBRP), focusing on non-draft documents. This ensures that only finalized billing data is considered in the reporting or processing logic.

8. \*\*Financial and Customer Data Integration\*\*

- Financial data is fetched from ACDOCA, and customer master data is retrieved from KNA1. This allows the program to enrich delivery and billing data with financial and customer dimensions, supporting comprehensive reporting or analysis.

9. \*\*Commodity Code and Classification Handling\*\*

- The program interacts with MARC and invokes specialized classes (/sapsll/cl\_mm\_cls\_service) to fetch commodity code classifications and details, supporting compliance and reporting requirements related to material classification.

10. \*\*Branch and Business Place Data Retrieval\*\*

- Data related to company codes and business places is extracted from P\_BusinessPlace, enabling organizational reporting and analysis.

11. \*\*Sales Document Status and Material Storage Data\*\*

- The program counts sales document headers (VBAK) and retrieves material storage location data from MARC and MARD, supporting inventory and sales status reporting.

12. \*\*Data Processing and Finalization\*\*

- The program processes the extracted data, performing operations such as substring extraction, conditional logic, and data movement between internal tables. It prepares a final dataset (IFINAL) by consolidating relevant fields from delivery, material, and plant data.

13. \*\*Salary and Amount Handling\*\*

- Specific subroutines handle the extraction and assignment of monetary values (e.g., salary from ACDOCA-DMBTR), supporting financial reporting or calculations.

14. \*\*Output Preparation\*\*

- After all data has been processed and consolidated, the program prepares the final output dataset, ready for reporting, further processing, or integration with downstream systems.

This functional approach ensures that the business requirement of extracting, validating, and consolidating data across multiple SAP modules (MM, SD, FI) is addressed in a structured, modular, and efficient manner, supporting both operational and analytical needs.

# 7. Functional Requirements

[Error: Missing output for section Functional Requirements]

# 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

[Error: Missing output for section Output]

# 10. UI Requirement

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Field Type | Default Value | Mandatory (Y/N) | Business Purpose / Validation Logic |
| P\_WERKS | Input | [To Be Filled] | [To Be Filled] | Single-value input for plant (WERKS) based on table T001W. Should validate against existing plant codes in T001W. |
| S\_MATNR | Range | [To Be Filled] | [To Be Filled] | Range input for material number (MATNR) based on table MARA. Should validate that entered material numbers exist in MARA. |

# 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. In the subroutine `fetch\_konv`, after the SELECT statement, there is a check for `sy-subrc = 0` to determine if data was successfully fetched. However, no explicit error handling or user notification is implemented in the provided code.

2. The subroutine `fetch\_and\_check\_plant` checks if the input parameter `p\_werks` is provided before performing a SELECT. No explicit error message or notification is triggered if the parameter is missing or if the SELECT fails.

3. In all other subroutines and methods, there is no explicit error handling, messaging, or user notification logic present.

4. No user-facing messages, pop-ups, or logs are defined for error or success scenarios in the provided code.

5. There are no requirements or business rules in the payload specifying error handling or notification mechanisms.

# 13. Assumptions & Dependencies

1. It is assumed that the user will provide valid input for the plant (`p\_werks`) and material number range (`s\_matnr`) on the selection screen.

2. The program depends on the existence and accessibility of standard SAP tables: T001W, MARA, LIPS, BSEG, VBRK, VBRP, ACDOCA, KNA1, PRCD\_ELEMENTS, MARC, MARD, VBAK, and P\_BusinessPlace.

3. The program assumes that the structures and fields referenced in the code (e.g., KONV, LIPS, MARA, DMBTR, MATNR, etc.) exist and are consistent with SAP standard definitions.

4. The program relies on the availability of the class `/sapsll/cl\_mm\_cls\_service` and its methods for commodity code classification.

5. The code assumes that the includes (`ZRCOPY\_SAMPLE\_ECC\_CODE\_TOP\_V1`, `ZRCOPY\_SAMPLE\_ECC\_CODE\_F01\_V1`, etc.) are present and correctly contain the necessary declarations and subroutine implementations.

6. No explicit error handling or authorization checks are implemented in the provided code; it is assumed that users have the necessary permissions to execute the program and access the referenced tables.

7. The program assumes that the SAP system is configured to allow inline data declarations and modern ABAP syntax as used in the code snippets.

# 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 = valid plant code (e.g., '1000') | i\_t001w populated with plant data for '1000' | [To Be Filled] | [To Be Filled] |
| TC02 | Validate plant input with empty value | p\_werks = '' (initial/empty) | No SELECT on T001W; i\_t001w remains initial | [To Be Filled] | [To Be Filled] |
| TC03 | Fetch material data for single material | s\_matnr = single value (e.g., 'MAT001') | imara contains MARA data for 'MAT001' | [To Be Filled] | [To Be Filled] |
| TC04 | Fetch material data for range of materials | s\_matnr = range (e.g., 'MAT001' to 'MAT010') | imara contains MARA data for all materials in range | [To Be Filled] | [To Be Filled] |
| TC05 | Fetch delivery items for given materials and plant | p\_werks = '1000', s\_matnr = 'MAT001','MAT002' | ilips contains LIPS data for materials 'MAT001','MAT002' and plant '1000' | [To Be Filled] | [To Be Filled] |
| TC06 | Prepare final data from delivery items | ilips populated | ifinal contains vbeln, posnr, matnr, werks for each ilips entry; lv\_lines = count of entries | [To Be Filled] | [To Be Filled] |
| TC07 | Fetch KONV/prcd\_elements data | - | ikonv contains knumv values from prcd\_elements, sorted by knumv | [To Be Filled] | [To Be Filled] |
| TC08 | Fetch billing document header (VBRK) where draft is space | - | lv\_vbeln contains vbeln from VBRK where draft = space | [To Be Filled] | [To Be Filled] |
| TC09 | Fetch billing document items (VBRP) where draft is space | - | lt\_vbrk contains vbeln and posnr from VBRP where draft = space | [To Be Filled] | [To Be Filled] |
| TC10 | Fetch financial data from ACDOCA | - | lt\_data contains bukrs, gjahr, belnr, buzei, hkont, dmbtr, wrbtr, waers, budat from ACDOCA | [To Be Filled] | [To Be Filled] |
| TC11 | Fetch customer numbers from KNA1 | - | lt\_data contains kunnr from KNA1, ordered by kunnr | [To Be Filled] | [To Be Filled] |
| TC12 | Fetch MARC stawn and call commodity code services | - | ls\_marc contains stawn, expme; methods get\_commodity\_code\_cls and get\_commodity\_code\_details called | [To Be Filled] | [To Be Filled] |
| TC13 | Fetch condition counter from prcd\_elements | - | lv\_dzaehk contains condition\_counter from prcd\_elements | [To Be Filled] | [To Be Filled] |
| TC14 | Fetch business place data | - | lt\_data contains bukrs and branch from P\_BusinessPlace, ordered | [To Be Filled] | [To Be Filled] |
| TC15 | Fetch MARC and MARD data with substring logic | lv\_matnr = 'MAT1234567' | lt\_data contains MARC matnr matching substring; lv\_lsobs contains MARD lsobs | [To Be Filled] | [To Be Filled] |
| TC16 | Count entries in VBAK | - | lv\_vbak\_cnt contains count of VBAK entries | [To Be Filled] | [To Be Filled] |
| TC17 | Fetch material data with substring and conditional message | lv\_matnr, lv\_mtart set; lv\_matnr\_chk4 = '1234' | lt\_table contains MARA data; message variable populated with concatenated string | [To Be Filled] | [To Be Filled] |
| TC18 | Fetch single MARC material using substring | lv\_matnr = 'MAT1234567' | lv\_marc\_matnr contains MARC matnr matching substring | [To Be Filled] | [To Be Filled] |
| TC19 | Populate salary from ACDOCA | acdoca-dmbtr = 1000.00 | lv\_salary = 1000.00 | [To Be Filled] | [To Be Filled] |

# 15. Sign-Off

|  |  |  |  |
| --- | --- | --- | --- |
| Role | Name | Signature | Date |
| Prepared By | [To Be Filled] | [To Be Filled] | [To Be Filled] |
| Approved By | [To Be Filled] | [To Be Filled] | [To Be Filled] |
| Client Sign-Off | [To Be Filled] | [To Be Filled] | [To Be Filled] |

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