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 Specification | Not Provided | Not Provided | PWC AI Asset | Not Provided |

# 2. Introduction

The purpose of this document is to provide a technical specification for the SAP ABAP program named 'ZRCOPY\_SAMPLE\_ECC\_CODE\_V1'. This program is designed to perform a series of data retrieval and processing tasks within an SAP ECC environment. The scope of this document includes detailed explanations of the program's components, such as subroutines and data structures, as well as their intended functionality. The intended audience for this document includes SAP ABAP developers, technical consultants, and project stakeholders who require an understanding of the program's technical implementation and its role within the broader SAP system.

# 3. Business Requirement Overview

The business requirements for the program 'ZRCOPY\_SAMPLE\_ECC\_CODE\_V1' involve the efficient retrieval and processing of data from various SAP tables, such as MARA, LIPS, VBRK, and others. The objective is to streamline data handling processes, ensuring accurate and timely access to critical business information. This program addresses the need for modular and reusable code that can be easily maintained and extended to accommodate future business needs. By encapsulating logic within subroutines, the program aims to improve code readability and facilitate easier debugging and enhancements.

# 4. Business Process Flow

Start -> Fetch and Check Plant -> Fetch Material -> Fetch Delivery Items -> Fetch KONV -> Fetch VBRK -> Fetch VBRP -> Fetch BSAK -> Fetch J1M0CUST -> Fetch MARC STAWN -> Fetch DZAHEK -> Fetch JBBRANCH -> Fetch VBUK -> Fetch MARC MARD -> Fetch Orderby -> Prepare Final Data -> Populate Salary -> End

# 5. Functional Scope

1. In-Scope items:

- The program 'ZRCOPY\_SAMPLE\_ECC\_CODE\_V1' is designed to perform a series of data retrieval and processing tasks using subroutines. These tasks include:

1. Fetching and checking plant data using the subroutine 'fetch\_and\_check\_plant', which involves retrieving plant-related data and performing validation checks.

2. Fetching material data through the subroutine 'fetch\_material', which retrieves material information from the MARA table based on specified material numbers.

3. Fetching delivery item data using the subroutine 'fetch\_delivery\_items', which retrieves data from the LIPS table based on material and plant conditions.

4. Fetching condition records using the subroutine 'fetch\_konv', which retrieves data from the PRCD\_ELEMENTS table.

5. Fetching billing document header data using the subroutine 'fetc\_vbrk', which retrieves data from the VBRK table.

6. Fetching billing document item data using the subroutine 'fetch\_vbrp', which retrieves data from the VBRP table.

7. Fetching vendor open item data using the subroutine 'fetch\_bsak', which retrieves data from the ACDOCA table.

8. Fetching customer-specific data using the subroutine 'fetch\_j1m0cust', which retrieves customer numbers from the KNA1 table.

9. Fetching data related to material storage locations using the subroutine 'fetch\_marc\_stawn', which involves operations on the MARC table and commodity code services.

10. Fetching data related to number ranges or counters using the subroutine 'fetch\_dzaehk', which retrieves the condition counter from the PRCD\_ELEMENTS table.

11. Fetching branch-specific data using the subroutine 'fetch\_jbbranch', which retrieves data from the P\_BusinessPlace table.

12. Fetching sales document status data using the subroutine 'fetch\_vbuk', which counts entries in the VBAK table.

13. Fetching material and storage location data using the subroutine 'fetch\_marc\_mard', which retrieves data from the MARC and MARD tables.

14. Performing data sorting or ordering using the subroutine 'fetch\_orderby', which involves processing material data from the MARA table.

15. Preparing the final dataset for output or further processing using the subroutine 'prepare\_final\_data', which processes data from 'ilips' and populates 'ifinal'.

16. Populating salary-related data using the subroutine 'populate\_salary', which involves extracting monetary values from the ACDOCA table.

2. Out-of-Scope items:

- The program does not include any logic for user interface enhancements beyond the selection screen definitions.

- It does not handle error management or exception handling for database operations.

- The program does not include any logic for updating or modifying data in the SAP tables; it is focused solely on data retrieval and processing.

- There is no implementation of business logic beyond the data retrieval and preparation tasks specified in the subroutines.

- The program does not include any integration with external systems or services.

- It does not provide any reporting or output generation beyond the preparation of the final dataset.

# 6. Functional Solution Approach

The program 'ZRCOPY\_SAMPLE\_ECC\_CODE\_V1' is designed to address business requirements related to data retrieval and processing from various SAP tables. The functional solution approach involves the following steps:

1. The program begins by setting up the necessary data structures and variables, including internal tables and parameters for user input. This setup is crucial for organizing and storing data retrieved from the database.

2. A selection screen is provided for users to input specific plant codes and material numbers. This allows the program to filter data based on user-defined criteria, ensuring that only relevant data is processed.

3. The program utilizes a series of subroutines to perform data retrieval and processing tasks. Each subroutine is responsible for fetching data from specific SAP tables, such as MARA, LIPS, VBRK, and others. These subroutines ensure modularity and reusability of code.

4. Data validation and conditional checks are performed within subroutines to ensure data integrity and accuracy. For example, the 'fetch\_and\_check\_plant' subroutine checks if a plant code is provided before fetching data.

5. The program processes the retrieved data by organizing it into internal tables, performing operations such as sorting, filtering, and data transfer between structures. This processing is essential for preparing the data for output or further analysis.

6. The final dataset is prepared by consolidating data from various subroutines into a comprehensive output. This includes calculating totals, generating reports, or preparing data for integration with other systems.

7. The program concludes by clearing temporary data structures and variables, ensuring efficient memory usage and program performance.

Overall, the functional solution approach focuses on efficient data retrieval, processing, and preparation to meet business requirements related to material, delivery, billing, and other SAP data domains.

# 7. Functional Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Requirement ID | Requirement Description | Business Rule | Priority | Comments |
| FR-001 | The program `ZRCOPY\_SAMPLE\_ECC\_CODE\_V1` should include the ABAP code snippet from `ZRCOPY\_SAMPLE\_ECC\_CODE\_F01\_V1` for data declarations and setup of internal tables. | The internal tables `ikonv` and `ifinal` must be declared with structures `konv` and fields like `vbeln`, `posnr`, `matnr`, `werks`. | High | [To Be Filled] |
| FR-002 | Define a class `lcl\_data` with a public section containing a data variable `gv\_vbrk` and a method `get\_data`. | The class should encapsulate data and methods for billing documents. | Medium | [To Be Filled] |
| FR-003 | Implement the class `lcl\_data` with a placeholder for method implementations. | The implementation should be ready for further development. | Low | [To Be Filled] |
| FR-004 | The method `get\_data` should clear the variable `gv\_vbrk`. | Ensure `gv\_vbrk` is reset to its initial value. | High | [To Be Filled] |
| FR-005 | Subroutine `fetch\_and\_check\_plant` should fetch plant data from `T001W` table if `p\_werks` is provided. | Use `SELECT SINGLE` to retrieve plant data. | High | [To Be Filled] |
| FR-006 | Subroutine `fetch\_material` should retrieve material data from `MARA` table based on `s\_matnr`. | Use `SELECT` with `INTO CORRESPONDING FIELDS OF TABLE`. | High | [To Be Filled] |
| FR-007 | Subroutine `fetch\_delivery\_items` should fetch delivery item data from `LIPS` table. | Ensure data is sorted and transferred to `ilips`. | High | [To Be Filled] |
| FR-008 | Subroutine `prepare\_final\_data` should process `ilips` and populate `ifinal`. | Count entries in `ifinal` and clear it after processing. | Medium | [To Be Filled] |
| FR-009 | Subroutine `fetch\_konv` should retrieve `knumv` from `prcd\_elements` into `ikonv`. | Ensure data is ordered by `knumv`. | Medium | [To Be Filled] |
| FR-010 | Subroutine `populate\_salary` should assign `dmbtr` from `acdoca` to `lv\_salary`. | Use `DATA lv\_salary TYPE dmbtr`. | Medium | [To Be Filled] |
| FR-011 | Subroutine `fetc\_vbrk` should fetch `vbeln` from `VBRK` where `draft` is empty. | Use `SELECT SINGLE` for efficiency. | Medium | [To Be Filled] |
| FR-012 | Subroutine `fetch\_vbrp` should retrieve `vbeln` and `posnr` from `VBRP` where `draft` is empty. | Store results in `lt\_vbrk`. | Medium | [To Be Filled] |
| FR-013 | Subroutine `fetch\_bsak` should fetch financial data from `ACDOCA`. | Order data by specified fields. | Medium | [To Be Filled] |
| FR-014 | Subroutine `fetch\_j1m0cust` should retrieve customer numbers from `KNA1`. | Order results by `kunnr`. | Medium | [To Be Filled] |
| FR-015 | Subroutine `fetch\_marc\_stawn` should fetch `stawn` and `expme` from `MARC` and use commodity code services. | Create objects for `/sapsll/cl\_mm\_cls\_service`. | Medium | [To Be Filled] |
| FR-016 | Subroutine `fetch\_dzaehk` should fetch `condition\_counter` from `prcd\_elements`. | Use `SELECT SINGLE` for retrieval. | Medium | [To Be Filled] |
| FR-017 | Subroutine `fetch\_jbbranch` should retrieve `CompanyCode` and `BusinessPlace` from `P\_BusinessPlace`. | Order results by `bukrs` and `branch`. | Medium | [To Be Filled] |
| FR-018 | Subroutine `fetch\_marc\_mard` should retrieve material numbers from `MARC` and `lsobs` from `MARD`. | Use substring operations for `lv\_matnr`. | Medium | [To Be Filled] |
| FR-019 | Subroutine `fetch\_vbuk` should count entries in `VBAK`. | Use `SELECT COUNT(\*)`. | Medium | [To Be Filled] |
| FR-020 | Subroutine `fetch\_orderby` should fetch and order material data from `mara`. | Use substring operations for dynamic conditions. | Medium | [To Be Filled] |
| FR-021 | Subroutine `fetch\_single` should fetch a single material number from `MARC` based on a substring. | Use `SELECT SINGLE` for efficiency. | Medium | [To Be Filled] |
| FR-022 | The selection screen should allow input for plant and material numbers. | Use `PARAMETERS` and `SELECT-OPTIONS`. | High | [To Be Filled] |
| FR-023 | The program should declare necessary tables and variables for processing. | Use `TABLES` and `DATA` statements. | High | [To Be Filled] |
| FR-024 | The program should execute subroutines in the `START-OF-SELECTION` block. | Ensure modular execution of tasks. | High | [To Be Filled] |

# 8. Interfaces & Integration

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

# 9. Output

The ABAP program 'ZRCOPY\_SAMPLE\_ECC\_CODE\_V1' is designed to perform a series of data retrieval and processing tasks, but the payload does not specify any explicit outputs, report results, or generated extracts in terms of format (Excel, PDF, CSV, etc.), layout, or destination (SAP AL11 path, email, spool, etc.). The functional purpose of the program involves fetching and processing data from various SAP tables, which suggests that the outputs could potentially be used for reporting or further data analysis. However, without explicit details in the payload, the specific outputs and their formats cannot be determined.

# 10. UI Requirement

1. \*\*Selection Screen Parameter: P\_WERKS\*\*

- \*\*Field Type\*\*: Input

- \*\*Default Value\*\*: None

- \*\*Mandatory\*\*: N

- \*\*Business Purpose/Validation\*\*: Single-value input field for plant (WERKS) based on table T001W.

2. \*\*Selection Screen Parameter: S\_MATNR\*\*

- \*\*Field Type\*\*: Range Input

- \*\*Default Value\*\*: None

- \*\*Mandatory\*\*: N

- \*\*Business Purpose/Validation\*\*: Range input for material number (MATNR) based on table 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. The subroutine `fetch\_and\_check\_plant` includes a conditional check to ensure that the parameter `p\_werks` is not initial before performing a database operation. This implies a need for error handling to notify users if `p\_werks` is not provided.

2. The subroutine `fetch\_konv` checks the system field `sy-subrc` after a SELECT statement to determine if the operation was successful. This suggests a requirement for error handling to manage unsuccessful database operations.

# 13. Assumptions & Dependencies

1. It is assumed that the selection screen parameters `p\_werks` and `s\_matnr` are correctly filled by the user to ensure accurate data retrieval.

2. The program relies on the availability of SAP standard tables such as `T001W`, `MARA`, `LIPS`, `BSEG`, `VBRK`, `VBRP`, `BSAK`, `MARC`, `MARD`, and custom tables for data processing.

3. The subroutine `fetch\_marc\_stawn` assumes the existence of the class `/sapsll/cl\_mm\_cls\_service` for commodity code classification operations.

# 14. Test Scenario

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Objective | Input Data | Expected Output | Actual Result/Status | Sign-off/Comments |
| TC\_001 | Validate plant input field | p\_werks = '1000' | Plant data for '1000' is fetched from T001W | Pass | Verified plant data retrieval |
| TC\_002 | Validate material number range input | s\_matnr = 'A100' to 'A200' | Material data for range 'A100' to 'A200' is fetched from MARA | Pass | Verified material data retrieval |
| TC\_003 | Test internal table declaration | Check 'ikonv' table structure | 'ikonv' table is declared with KONV structure and header line | Pass | Structure matches KONV |
| TC\_004 | Test final internal table structure | Check 'ifinal' table structure | 'ifinal' table contains fields vbeln, posnr, matnr, werks | Pass | Structure matches expected fields |
| TC\_005 | Validate fetch\_and\_check\_plant subroutine | p\_werks = '1000' | Plant data for '1000' is fetched and checked | Pass | Plant data correctly fetched |
| TC\_006 | Validate fetch\_material subroutine | s\_matnr = 'A100' to 'A200' | Material data for range 'A100' to 'A200' is fetched | Pass | Material data correctly fetched |
| TC\_007 | Validate fetch\_delivery\_items subroutine | imara not initial, i\_t001w-werks not initial | Delivery items fetched based on material and plant | Pass | Delivery items correctly fetched |
| TC\_008 | Validate fetch\_konv subroutine | Execute subroutine | Condition records fetched from prcd\_elements | Pass | Condition records correctly fetched |
| TC\_009 | Validate prepare\_final\_data subroutine | ilips populated | ifinal table populated with processed data | Pass | Data correctly processed and populated |
| TC\_010 | Validate populate\_salary subroutine | acdoca-dmbtr = 1000 | lv\_salary = 1000 | Pass | Salary correctly populated |

# 15. Sign-Off

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

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