LA101R.MBR

Path: NXCLOUD/rpgsrc/LA101R.MBR **Generated**: 2026-01-08 15:43:00 **Processing Time**: 16969ms

Business Logic for LA101R

This document outlines the business rules that govern the maintenance of status codes within the system, based on an analysis of the RPG program LA101R. The primary focus is on how the program manages the retrieval and updating of inventory status codes.

The core logic for maintaining status codes is contained within the A1BLD subroutine in LA101R. The program processes records from the lstsl1 and lstslu files to retrieve and update inventory status code information based on user input.

Order Status and Header Rules

LA101R: lstsl1, lstslu

1. Retrieve Status Code Information

- **Logic:** The program retrieves status code details from the lstsl1 file based on a key. If a record is found, it populates various fields with the retrieved values; otherwise, it sets the fields to blank or zero.
- **File:** lstsl1 (Inventory Status Codes)
- **Field:** lstsl1_key
- **Condition:** The process will not select a record if the key does not match any existing records in the lstsl1 file.

2. Display Status Code Information

- **Logic:** After retrieving the status code information, the program displays the information on the screen for user interaction.
- **File:** a1bld (Display Format for Status Codes)
- **Field:** Various fields representing status code attributes
- **Condition:** The display occurs after the retrieval of information, and if the user does not press the exit key (F3).

Configuration and Authorization Rules

1. Update Status Code Information

- **Logic:** The program updates the lstslu file with the values entered by the user. If the record exists, it updates the existing record; if not, it creates a new record.
- **Files:**
- lstslu (Updated Inventory Status Codes)
- **Fields:**
- laabrk (Status Code)
- laabin (Additional Information)
- **Condition:** The update occurs only if the record is found in the lstslu file; otherwise, a new record is written.

2. Initialize Program Variables

•**Logic:** The program initializes local variables and sets the firm identifier for processing.
•**File:** None
•**Field:** w_firm
•**Condition:** This initialization occurs at the start of the program to prepare for data processing.

Financial and Transactional Rules

1. Log User Actions

•**Logic:** The program logs user actions by capturing the user identifier and timestamps whenever a record is updated or created.
•**File:** lstslu (Updated Inventory Status Codes)
•**Fields:**
•laaeus (User Identifier)
•laaeda (Date of Action)
•**Condition:** This logging occurs during the update or write process of the inventory status code.

2. Handle Non-Existent Records

•**Logic:** If the status code does not exist in the lstsl1 file, the program sets all relevant fields to default values (blank or zero).
•**File:** lstsl1 (Inventory Status Codes)
•**Condition:** This handling occurs when the initial chain operation does not find a matching record.

Special Conditions (Program-Specific)

1. End Program Logic (LA101R)

•**Logic:** The program concludes by setting the last record indicator and returning control to the calling program.
•**File:** None
•**Field:** *inlr
•**Condition:** This logic executes when the user opts to exit the program.

2. Subroutine for Initialization (LA101R)

•**Logic:** The subroutine initializes keys for reading and updating the inventory status codes, ensuring that the firm identifier is included in the keys.
•**File:** None
•**Fields:** lstsl1_key, lstslu_key
•**Condition:** This subroutine runs at the beginning of the program to prepare for data access.

Subprogram Calls Affecting Logic

Beyond direct file checks, several external subprograms are called that play a significant role in the workflow.

1. A1BLD (Display Subroutine)

•**Trigger:** Called after retrieving status code information to display it to the user.
•**Logic:** This subroutine formats and presents the retrieved status code information for user interaction.
•**Impact:** This call acts as a **major logical gateway** for user input and interaction.

2. *inzsr (Initialization Subroutine)

- **Trigger:** Called at the beginning of the program to set up necessary keys and variables.
- **Logic:** This subroutine initializes the keys for reading and updating records in the inventory status code files.
- **Impact:** This is a **critical setup step** that ensures the program operates with the correct context.

3. end_pgm (End Program Logic)

- **Trigger:** Called when the user chooses to exit the program.
- **Logic:** This subroutine finalizes the program state and cleans up any resources.
- **Impact:** Represents the conclusion of the business logic processing for this program, ensuring all operations are finalized correctly.