

# LA101R.MBR

**Path:** NXCLOUD/rpgsrc/LA101R.MBR **Generated:** 2026-01-09 10:13:44 **Processing Time:** 14189ms

## 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 interacts with the inventory codes and updates their statuses.

The core logic for maintaining the status codes is contained within the A1BLD subroutine in LA101R. The program retrieves information from the lstsl1 file, processes it, and updates the lstslu file accordingly.

## Order Status and Header Rules

LA101R: lstsl1, lstslu

### 1. Retrieve Inventory Code Information

- Logic:** The program attempts to retrieve the inventory code details from the lstsl1 file based on a key. If found, it populates various fields with the retrieved values; if not found, it initializes the fields to blank or zero.
- File:** lstsl1 (Inventory Code File)
- Field:** lstsl1\_key
- Condition:** The process will not select a record if lstsl1\_key is not found.

### 2. Display Inventory Code Details

- Logic:** After retrieving the inventory code information, the program displays the details on the screen for user interaction.
- File:** a1bld (Display Format)
- Field:** Various fields populated from lstsl1
- Condition:** The display occurs after the retrieval of inventory code information.

## Configuration and Authorization Rules

### 1. Update Inventory Code

- Logic:** The program updates the lstslu file with the values from the user input fields if the inventory code exists; otherwise, it creates a new record.
- Files:**
  - lstslu (Updated Inventory Code File)
  - lstsl1 (Original Inventory Code File)
- Fields:**
  - laabr (from lstslu)
  - a1abr (from user input)
- Condition:** The update occurs only if the lstslu\_key is found; otherwise, a new record is written.

### 2. End Program Logic

- Logic:** The program checks for the F3 key input to terminate the program gracefully.
- File:** None

- Field:** None
- Condition:** The program will end if the F3 key is pressed.

## Financial and Transactional Rules

### 1. Track Update Timestamps

- Logic:** When updating an existing inventory code, the program records the timestamps for when the record was updated.
- File:** Istslu (Updated Inventory Code File)
- Fields:**
  - laaeda (Update Date)
  - laaeti (Update Time)
- Condition:** Timestamps are recorded only if the inventory code exists.

### 2. Initialize New Record Fields

- Logic:** If the inventory code does not exist, the program initializes several fields to default values (e.g., blanks or zeros) for a new record.
- File:** Istslu (New Inventory Code File)
- Condition:** This initialization occurs when the Istslu\_key is not found.

## Special Conditions (Program-Specific)

### 1. User Identification (LA101R)

- Logic:** The program captures the user ID from a local data area for tracking changes made to inventory codes.
- File:** None
- Field:** l\_user
- Condition:** This information is used during the update process to log who made the changes.

### 2. Firm Identification (LA101R)

- Logic:** The program ensures that the firm identifier is included in the keys for both reading and updating inventory codes.
- File:** None
- Fields:** w\_firm (Firm Identifier)
- Condition:** The firm identifier is set at the beginning of the program and is used for all subsequent operations.

## 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:** This subroutine is called after retrieving the inventory code information.
- Logic:** It formats and displays the inventory code details for user interaction.
- Impact:** This call acts as a **major logical gateway** for user input and interaction.

### 2. End Program Logic

- Trigger:** This logic is executed upon the program's termination.

- Logic:** It ensures that all resources are released and the program ends cleanly.
- Impact:** This is a **destructive filtering step** that ensures no lingering processes remain.

### 3. Initialization Subroutine

- Trigger:** This subroutine is called at the beginning of the program.
- Logic:** It sets up the necessary keys and initializes variables for processing.
- Impact:** This represents the handoff to the next major business function, ensuring the program starts with the correct context.