

# RC100R.MBR

**Path:** NXCLOUD/rpgsrc/RC100R.MBR **Generated:** 2026-01-08 15:43:00 **Processing Time:** 13174ms

## Business Logic for RC101R

This document outlines the business rules that govern the maintenance of fixed data for debt collection, based on an analysis of the RPG program RC101R. The primary focus is on the logic that handles user interactions with subfiles, data validation, and updates to the database.

The core logic for maintaining fixed data is contained within the \*inzsr subroutine in RC101R. This program processes user inputs, manages subfile displays, and updates records in the database.

## Order Status and Header Rules

RC101R: RCA0PF, RCA0LR, RCA0LU

### 1. Subfile Initialization

- Logic:** Initializes the subfile for displaying records related to debt collection agencies.
- File:** RCA0PF (Debt Collection Agency Data)
- Field:** rca0l1\_0fir
- Condition:** The subfile is populated with records from RCA0PF based on the current firm.

### 2. Record Validation

- Logic:** Validates that the selected debt collection agency exists before allowing updates.
- File:** RCA0LR (Debt Collection Agency Records)
- Field:** rca0lr\_0fir
- Condition:** The process will not proceed if the agency identifier does not match any existing records.

## Configuration and Authorization Rules

### 1. User Firm Initialization

- Logic:** Sets the user's firm for processing and initializes the display.
- Files:**
  - RCA0PF (Debt Collection Agency Data)
  - RCA0LR (Debt Collection Agency Records)
- Fields:**
  - l\_firm (User's Firm)
  - rca0l1\_0fir (Agency Identifier)
- Condition:** The program initializes with the user's firm from the local data area.

### 2. Function Key Processing

- Logic:** Handles function key inputs for various operations (create, update, delete).
- File:** RCA0PF (Debt Collection Agency Data)
- Field:** b1valg
- Condition:** The program will execute different subroutines based on the function key pressed.

## Financial and Transactional Rules

## 1. Record Update Logic

- Logic:** Updates or inserts records into the RCA0LR file based on user input.
- File:** RCA0LR (Debt Collection Agency Records)
- Fields:**
  - rc0txt (Text Description)
  - rc0kto (Account Number)
- Condition:** The record is updated if it exists; otherwise, a new record is created.

## 2. Record Deletion Logic

- Logic:** Deletes a record from the RCA0LR file upon user confirmation.
- File:** RCA0LR (Debt Collection Agency Records)
- Condition:** The record will be deleted if the user confirms the deletion action.

# Special Conditions (Program-Specific)

## 1. Duplicate Record Handling (RC101R)

- Logic:** Displays a message if the user attempts to create a record that already exists.
- File:** RCA0LR (Debt Collection Agency Records)
- Field:** rca0lr\_0fir
- Condition:** If a record with the same identifier is found, the program will show an error message.

## 2. Agency Validation (RC101R)

- Logic:** Validates the selected debt collection agency against known agencies.
- File:** RCA0PF (Debt Collection Agency Data)
- Fields:** c2inkf (Agency Identifier)
- Condition:** The program checks if the agency identifier is valid; if not, it displays an error message.

# Subprogram Calls Affecting Logic

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

## 1. AA005R (Error Message Display)

- Trigger:** Called when an unknown debt collection agency is selected.
- Logic:** Displays an error message to the user.
- Impact:** This call acts as a **validation step** to ensure that only recognized agencies are processed.

## 2. xc2bld (Record Maintenance Subroutine)

- Trigger:** Called when creating or updating a record.
- Logic:** Handles the logic for maintaining records in the RCA0LR file.
- Impact:** This is a **critical operation** that ensures data integrity during updates.

## 3. xd1win (Delete Confirmation Window)

- Trigger:** Called when the user opts to delete a record.
- Logic:** Displays a confirmation window for deletion.
- Impact:** This represents a **safeguard** against accidental deletions, ensuring user confirmation before proceeding.

