RS001R.MBR

Path: NXCLOUD/rpgsrc/RS001R.MBR **Generated**: 2026-01-08 15:04:25 **Processing Time**: 13275ms

Business Logic for Customer Number Retrieval

This document outlines the business rules that govern the retrieval of the last used customer number, based on an analysis of the RPG program RS001R. The primary focus is on how the program processes customer numbers and checks for validity.

The core logic for retrieving the last used customer number is contained within the main processing logic of the program. The program checks various conditions and updates the customer number accordingly.

Customer Number Retrieval Rules

Customer Number Retrieval Process: raa4l1, rkunlrr, raa1l1r, raa4lur

1. Check for Last Used Customer Number

- **Logic:** The program attempts to retrieve the last used customer number from the raa4l1 file. If not found, it sets a return code to indicate failure.
- **File:** raa4l1 (Last used customer number file)
- **Field:** ra4knr
- **Condition:** The process will not proceed if the record is not found (indicated by *in90 being on).

2. Update Last Used Customer Number

- **Logic:** If a valid last used customer number is found, the program updates it based on specific conditions (incrementing, decrementing, or setting to a specific value).
- **File:** raa4l1 (Last used customer number file)
- **Field:** ra4knr
- **Condition:** The update occurs based on the value of w_test and the current customer number.

Boundary and Existence Check Rules

1. Check Against Boundaries

- **Logic:** The program checks if the updated customer number is within specified limits. If it exceeds these limits, it sets an error return code.
- **Files:**
- raa1l1 (Boundary limits file)
- **Fields:**
- ra1hkt (Lower boundary)
- ra1hrs (Upper boundary)
- **Condition:** The process will set the return code to 'N' if the updated customer number is less than or equal to the lower boundary or greater than or equal to the upper boundary.

2. Check for Customer Number Existence

- **Logic:** The program checks if the last used customer number exists in the customer register. If not, it increments the number until a valid one is found.
- **File:** rkunlr (Customer register)

- **Field:** rkkund
- **Condition:** The process will continue to increment w_kund until a valid customer number is found (indicated by *in90 being off).

Update and Finalization Rules

1. Final Update of Customer Number

- **Logic:** After determining a valid customer number, the program updates the last used customer number in the raa4lur file.
- **File:** raa4lur (Last used customer number update file)
- **Field:** ra4knr
- **Condition:** The update occurs if the customer number is valid and the record is found.

2. Return Code Handling

- **Logic:** The program sets the return code to indicate success or failure based on the operations performed.
- **File:** N/A
- **Condition:** The return code is set to 'N' for failure or blank for success before exiting the program.

Subprogram Calls Affecting Logic

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

1. *entry (Entry Point for Program)

- **Trigger:** This program is called with parameters for processing.
- **Logic:** It initializes the parameters and sets up the keys for file access.
- **Impact:** This call acts as a **major logical entry point** for the customer number retrieval process.

2. *inzsr (Initialization Subroutine)

- **Trigger:** This subroutine is invoked at the beginning of the program.
- **Logic:** It initializes the program and prepares the keys for file access.
- **Impact:** This is a **critical setup step** that ensures the program has the necessary context for processing.

3. chain (File Access Logic)

- **Trigger:** The program uses chain operations to access records in various files.
- **Logic:** It retrieves records based on the keys defined in the program.
- **Impact:** This represents the **core data retrieval mechanism** that drives the entire customer number processing logic.