AA000R.MBR

Path: NXCLOUD/rpgsrc/AA000R.MBR **Generated**: 2026-01-08 12:06:00 **Processing Time**: 10404ms

Business Logic for User Registration Lookup

This document outlines the business rules that govern the user registration lookup process, based on an analysis of the RPG program AA000R. The primary focus is on how user records are accessed and validated within the system.

The core logic for user record retrieval is contained within the main processing logic of the AA000R program. The program checks for the existence of a user in the user register and validates specific conditions based on the user input.

User Record Lookup Rules

User Registration: ausrl1

1. User Record Retrieval

- **Logic:** The program attempts to retrieve a user record from the ausrl1 file using the provided user identifier. If the record is not found, it sets the status flags to indicate failure.
- **File:** ausrl1 (User Register)
- **Field:** ausrl1_user
- **Condition:** The process will not proceed if the user record is not found (*in60 is set to *off).

2. User Validation for Specific Cases

- **Logic:** The program checks if the user belongs to specific predefined groups or if the first character of the username is 'Q'. If so, it sets the status flags to indicate that the user is not valid.
- **File:** ausrl1 (User Register)
- **Field:** p_user
- **Condition:** The process will not proceed if p_user is equal to 'ASPKASSE', 'NORGROS', 'ASPRMI', or if the first character of p_user is 'Q'.

Initialization and Parameter Handling Rules

1. Program Initialization

- **Logic:** The program initializes parameters for the user lookup process at the beginning of execution, setting the p_okok flag to zero.
- **Files:** None
- **Fields:** p_okok
- **Condition:** This initialization occurs every time the program is called.

2. Parameter Passing

- **Logic:** The program accepts parameters for job ID, user identifier, and a status flag, which are used throughout the processing logic.
- **File:** None
- **Fields:** p_jobb, p_user, p_okok
- **Condition:** These parameters are required for the program to function correctly.

Special Conditions (Program-Specific)

1. User from Webshop (AA000R)

- **Logic:** If the user is identified as coming from a webshop, the program does not require the user to be defined in the user register.
- **File:** ausrl1 (User Register)
- **Field:** p_user
- **Condition:** This condition is checked to allow users from specific sources to bypass standard validation checks.

2. Environment Variable Handling (AA000R)

- **Logic:** The program includes a variable for the environment (e.g., A01, A02) that may influence processing but is not explicitly detailed in the logic.
- **File:** None
- **Fields:** dsmilj
- **Condition:** This variable is set based on the environment and may affect other processing logic.

Subprogram Calls Affecting Logic

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

1. *inzsr (Initialization Subroutine)

- **Trigger:** This subroutine is called at the start of the program.
- **Logic:** It initializes the program's parameters, particularly setting the p_okok flag to zero.
- **Impact:** This initialization is crucial for ensuring that the program starts with a clean state.

2. chain (File Access Logic)

- **Trigger:** The chain operation is executed to retrieve the user record from the ausrl1 file.
- **Logic:** It attempts to locate the user based on the provided key.
- **Impact:** This operation is essential for determining whether the user exists in the system and affects the subsequent validation logic.

3. eval (Variable Assignments)

- **Trigger:** Throughout the program, the eval operation is used to assign values to various fields based on the retrieved user record.
- **Logic:** It populates local data structures with user information if the record is found.
- **Impact:** This step is critical for ensuring that the program has the necessary data to proceed with further processing.