AA000R.MBR

Path: NXCLOUD/rpgsrc/AA000R.MBR **Generated**: 2026-01-08 13:31:56 **Processing Time**: 12619ms

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 the program retrieves and validates user records from the user register.

The core logic for user record retrieval is contained within the main processing logic of the program, which includes reading from the user file and validating the user input. The program checks for the existence of a user record and applies specific business rules based on the user type.

User Record Retrieval Rules

User Registration Lookup: ausrl1

1. User Record Existence Check

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

2. User Type Validation

- **Logic:** The program checks if the user belongs to a specific set of predefined user types that are not allowed. If the user type matches any of these, it sets the status flags to indicate failure.
- **File:** ausrl1 (User registration file)
- **Field:** ausrl1_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:** Upon program initialization, the program sets the p_okok flag to indicate that the process has not yet validated the user.
- **Files:** None
- **Fields:** p_okok
- **Condition:** This initialization occurs at the beginning of the program in the initialization subroutine.

2. Parameter Passing

- **Logic:** The program accepts parameters for user identification and job context, which are used throughout the processing logic.
- **Files:** None
- **Fields:** p_user, p_jobb

•**Condition:** These parameters are mandatory for the program to function correctly and are passed at the program entry point.

_

Special Conditions (Program-Specific)

1. User from Webshop Check (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 registration file)

•**Field:** p_user

•**Condition:** This check is applied if the user identifier starts with 'ASPKASSE', 'NORGROS', 'ASPRMI', or if the first character of p_user is 'Q'.

2. Environment Variable Handling (AA000R)

•**Logic:** The program includes a variable to handle different environments (e.g., A01, A02) that may affect processing logic.

•**File:** None

•**Fields:** dsmilj

•**Condition:** This variable is utilized to adapt the program's behavior based on the operational environment.

_

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 beginning of the program execution.

•**Logic:** It initializes the program's parameters and sets the p_okok flag to zero.

•**Impact:** This call ensures that the program starts with a clean state and prepares the necessary parameters for further processing.

2. chain (File Access)

•**Trigger:** This operation is invoked to access the user record based on the user identifier.

•**Logic:** It attempts to retrieve the user record from the ausrl1 file.

•**Impact:** This is a critical step that determines whether the user exists in the system and whether subsequent processing can continue.

3. eval (Variable Assignments)

•**Trigger:** This operation is used throughout the program to assign values to various variables based on the retrieved user record.

•**Logic:** It populates local data structures with user information if the record is found.

•**Impact:** This step is essential for ensuring that the program has the necessary data to perform its functions correctly.