AA000R.MBR

Path: NXCLOUD/rpgsrc/AA000R.MBR **Generated**: 2026-01-08 12:57:54 **Processing Time**: 10852ms

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 registration lookup is contained within the main program logic of AA000R. This program retrieves user records from the ausrl1 file and performs validation checks based on the user input.

User Record Access and Validation Rules

User Registration Lookup: ausrl1

1. User Record Retrieval

- **Logic:** The program attempts to retrieve a user record based on the provided user identifier. If the record is found, various parameters are populated with user data.
- **File:** ausrl1 (User registration file)
- **Field:** ausrl1_user
- **Condition:** The record is retrieved using a chain operation. If the record is not found, indicated by *in60, the program sets p_okok and dsokok to 0.

2. User Validation for Specific Cases

- **Logic:** The program checks if the user identifier matches specific predefined values or if the first character of the user identifier is 'Q'. If any of these conditions are met, the user is deemed invalid.
- **File:** ausrl1 (User registration file)
- **Field:** p_user
- **Condition:** The process will not accept a record 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 entry, the initialization subroutine sets the p_okok parameter to 0, indicating that the process has not yet validated the user.
- **Files:** None
- **Fields:** p_okok
- **Condition:** This logic is executed at the beginning of the program to prepare for user validation.

2. Parameter Passing

- **Logic:** The program receives parameters for user validation, including the job identifier and user identifier, which are critical for processing.
- **File:** None
- **Fields:** p_user, p_jobb

•**Condition:** These parameters are passed into the program during the call and are essential for subsequent processing.

Special Conditions (Program-Specific)

1. User Record Not Found Handling (AA000R)

•**Logic:** If the user record is not found, the program sets the validation flags (p_okok and dsokok) to 0, indicating that the user is not valid.

•**File:** ausrl1 (User registration file)

•**Field:** ausrl1_user

•**Condition:** This occurs when the chain operation does not find a matching record.

2. User from Webshop Handling (AA000R)

•**Logic:** The program allows users from specific webshops to bypass the validation check, meaning they do not need to be defined in the user register.

•**File:** ausrl1 (User registration file)

•**Fields:** p_user

•**Condition:** This condition is checked after the user record retrieval, allowing certain users to proceed without a defined record.

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.

•**Logic:** It initializes the p_okok parameter to 0 to indicate that the user has not yet been validated.

•**Impact:** This setup is crucial for ensuring that the program starts with a clean state for user validation.

2. chain Operation (User Record Access)

•**Trigger:** The chain operation is executed to retrieve the user record based on the ausrl1_user key.

•**Logic:** It attempts to find a matching user record in the ausrl1 file.

•**Impact:** This operation is essential for determining whether the user exists and subsequently affects the validation logic.

3. return Statement (Program Termination)

•**Trigger:** The program reaches its end and prepares to terminate.

•**Logic:** It sets the last record indicator (*inlr) to on, signaling the end of processing.

•**Impact:** This ensures that all resources are released and the program exits cleanly after processing the user lookup.