AA000R.MBR

Path: NXCLOUD/rpgsrc/AA000R.MBR **Generated**: 2026-01-08 15:42:59 **Processing Time**: 15035ms

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 the logic that retrieves user information from the user register file.

The core logic for user retrieval is contained within the main processing section of the program, which checks the user input against the user register file and sets various flags and data fields based on the results of this lookup.

User Lookup and Validation Rules

User Registration Lookup: ausrl1

1. User Record Retrieval

- **Logic:** The program retrieves a user record from the user register file (ausrl1) based on the provided user identifier (p_user). If the record is found, various data fields are populated with user-specific information.
- **File:** ausrl1 (User register file)
- **Field:** ausrl1_user
- **Condition:** The process will not proceed if the user record is not found (*in60 is set to *off).

2. User Validation

- **Logic:** The program checks if the user is one of the predefined special users (e.g., 'ASPKASSE', 'NORGROS', 'ASPRMI') 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 register file)
- **Field:** p_user
- **Condition:** The process will set p_okok and dsokok to 0 if the user is invalid.

Initialization and Parameter Handling Rules

1. Initialization of Parameters

- **Logic:** Upon program entry, the parameters p_okok, p_user, and p_jobb are initialized. The p_okok flag is set to 0, indicating that the user lookup has not yet succeeded.
- **Files:** None
- **Fields:** p_okok, p_user, p_jobb
- **Condition:** This initialization occurs at the beginning of the program execution.

2. Setting User Data Fields

- **Logic:** If the user record is successfully retrieved, various data fields are populated with values from the user record, including user ID, menu settings, and other associated attributes.
- **File:** ausrl1 (User register file)
- **Fields:** dswsid, dsuser, dsmeny, dssfil, dssbib, dssfir, dssavd, dsfnav, dsbfil, dsbfir, dsbavd, dslevl, dslagr, dsmilj

•**Condition:** This occurs only if the user record is found (*in60 is set to *off).

Special Conditions (Program-Specific)

1. Special User Conditions (AA000R)

•**Logic:** The program includes specific checks for certain users that do not require a defined user record, allowing access without a valid entry in the user register.

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

•**Field:** p_user

•**Condition:** The program allows users like 'ASPKASSE', 'NORGROS', 'ASPRMI', or those whose identifier starts with 'Q' to bypass the user validation process.

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 parameters and sets the p_okok flag to 0.

•**Impact:** This step ensures that the program starts with a clean state for processing user data.

2. chain (File Access Call)

•**Trigger:** The chain operation is invoked to retrieve the user record based on the user identifier.

•**Logic:** This operation attempts to locate the user record in the ausrl1 file.

•**Impact:** The success or failure of this operation directly influences the flow of logic in the program, determining whether user data is populated or if the process is halted.

3. return (Program Termination)

•**Trigger:** This statement is executed at the end of the program.

•**Logic:** It signals the end of the program execution and returns control to the calling program or environment.

•**Impact:** This is a crucial step for resource management and ensuring that the program exits cleanly.