AA000R.MBR

Path: NXCLOUD/rpgsrc/AA000R.MBR **Generated**: 2026-01-08 12:21:11 **Processing Time**: 11414ms

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 processing logic of the program, which retrieves user records from the ausrl1 file. The program checks for the existence of the user and validates certain conditions before proceeding.

User Record Validation 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 is found, it sets various data fields with the user's information.
- **File:** ausrl1 (User Registration File)
- **Field:** ausrl1_user
- **Condition:** The record will not be found if the user identifier does not match any entry in the ausrl1 file.

2. User Validation for Specific Conditions

- **Logic:** The program checks if the user identifier matches specific values (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 considered invalid.
- **File:** ausrl1 (User Registration File)
- **Field:** p_user
- **Condition:** The process will not validate the user if p_user equals 'ASPKASSE', 'NORGROS', 'ASPRMI', or if the first character of p_user is 'Q'.

Initialization and Parameter Rules

1. Initialization of Parameters

- **Logic:** Upon program initialization, the program sets the p_okok parameter to 0, indicating that the user lookup has not yet been validated.
- **Files:**
- ausrl1 (User Registration File)
- **Fields:**
- p_okok (Status of user validation)
- **Condition:** This initialization occurs at the beginning of the program execution.

2. Setting User Information on Successful Lookup

- **Logic:** If a user record is successfully retrieved, various fields are populated with the user's details, including user ID, menu, and other related information.

•**File:** ausrl1 (User Registration File)
•**Field:** dswsid, dsuser, dsmeny, dssfil, dssbib, dssfir, dssavd, dsfnav, dsbfil, dsbfir, dsbavd, dslevl , dslagr, dsmilj
•**Condition:** This assignment occurs only if the user record is found (i.e., *in60 is off).

Special Conditions (Program-Specific)

1. User from Online Store Check (AA000R)
•**Logic:** The program allows users from specific online stores to bypass the user definition requirement, setting p_okok to 0 if the user identifier matches certain predefined values.
•**File:** ausrl1 (User Registration File)
•**Field:** p_user
•**Condition:** This check is applied if the user identifier is one of the specified values or if the first character is 'Q'.
2. Program Termination Logic (AA000R)
•**Logic:** The program sets the last record indicator (*inlr) to on, indicating that the program is terminating.
•**File:** None
•**Condition:** This occurs at the end of the program, regardless of the outcome of the user lookup.

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 and sets up the parameter list for the program.
•**Impact:** This subroutine ensures that the program starts with a clean state for user validation.
2. chain (File Access Logic)
•**Trigger:** The chain operation is executed to retrieve the user record.
•**Logic:** It attempts to find the user record in the ausrl1 file based on the user identifier.
•**Impact:** This operation is critical as it determines whether the user exists and whether subsequent logic can proceed.
3. return (Program Exit Logic)
•**Trigger:** This is called at the end of the program.
•**Logic:** It finalizes the program execution and returns control to the calling program or environment.
•**Impact:** This represents the conclusion of the user lookup process, ensuring that all necessary cleanup is performed.