RY820R.MBR

Path: NXCLOUD/rpgsrc/RY820R.MBR **Generated**: 2026-01-08 15:43:00 **Processing Time**: 13071ms

Business Logic for Updating Match Register

This document outlines the business rules that govern the process of updating the match register, based on an analysis of the RPG program RY820R. The primary focus is on the logic that updates product information from the product register into the match register.

The core logic for updating the match register is contained within the main program flow of RY820R. The program reads through the match register and updates items with information from the product register.

Match Register and Product Rules

Updating Match Register: fjmatl1, fjvarl1, fjmatlu

1. Check for Blank Product Code

•**Logic:** The program checks if the product code (jovar2) is blank. If it is, the program skips to the next item in the match register.

•**File:** fjmatl1 (Match Register)

•**Field:** jovar2

•**Condition:** The process will skip updating if jovar2 is equal to *blank.

2. Update Information from Product Register

•**Logic:** If the product code is not blank, the program retrieves the product information from the product register (fjvarl1). If the product is found and the old value (joldo2) is different from the new value (jvldor), it updates the match register with the new product information.

•**File:** fjvarl1 (Product Register)

•**Field:** jvarl1_vare

•**Condition:** The process will update the match register if the product is found and the old value is not equal to the new value.

Key Variable Initialization Rules

1. Initialize Firm Key

•**Logic:** The program initializes the firm key (w_firm) from the local data area (l_firm) to be used for lookups in the match register and product register.

•**Files:**

•fjmatl1 (Match Register)

•fjvarl1 (Product Register)

•**Fields:**

•w_firm (Firm Key)

•**Condition:** This initialization occurs at the beginning of the program to ensure the correct firm context is used for all subsequent operations.

2. Key Definitions for Lookups

•**Logic:** The program defines keys for reading the match register and product register to ensure efficient access to the necessary records.

•**Files:**
•fjmatl1 (Match Register)
•fjvarl1 (Product Register)
•**Fields:**
•jmatl1_key (Key for Match Register)
•jvarl1_key (Key for Product Register)
•**Condition:** These keys are defined at the program's initialization to facilitate record access during processing.

Special Conditions (Program-Specific)

1. End of Program Logic
•**Logic:** At the end of the program, the *inlr indicator is set to *on, indicating that the program has completed processing and can release resources.
•**File:** N/A
•**Condition:** This is a standard practice in RPG programs to ensure proper cleanup after execution.

Subprogram Calls Affecting Logic

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

1. Initialization Subroutine
•**Trigger:** This subroutine is called at the beginning of the program.
•**Logic:** It initializes keys for reading and updating the match register and product register, as well as setting the firm context.
•**Impact:** This subroutine ensures that the program has the necessary context and keys ready for processing, acting as a foundational setup for the main logic.
2. Main Processing Loop
•**Trigger:** The main processing logic iterates through the match register.
•**Logic:** It reads each record, checks for conditions, and updates the match register based on the product register.
•**Impact:** This loop represents the core functionality of the program, executing the primary business logic of updating the match register with product information.
3. End of Program Logic
•**Trigger:** This logic executes once all records have been processed.
•**Logic:** It sets the last record indicator and returns control to the calling program or environment.
•**Impact:** This step is crucial for resource management and ensuring that the program concludes correctly without leaving open resources.