AF010R.MBR

Path: NXCLOUD/rpgsrc/AF010R.MBR **Generated**: 2026-01-08 12:57:56 **Processing Time**: 15867ms

Business Logic for Postcode Register Maintenance

This document outlines the business rules that govern the maintenance of the postcode register, based on an analysis of the RPG program AF010R. The primary focus is on the logic for handling postcode entries, including creation, updating, deletion, and display of records.

The core logic for managing the postcode register is contained within the various subroutines in AF010R. The program utilizes subfiles to display and manage records, allowing users to interact with postcode entries through a user-friendly interface.

Order Status and Header Rules

Postcode Register: aposl1, aposl2, aposlr, aposlu

1. Record Creation

•**Logic:** New postcode records can be created based on user input. If the postcode already exists, an error message is displayed.

•**File:** aposl1 (Postcode file)

•**Field:** aposl1_ponr

•**Condition:** The process will not create a record if aposl1_ponr already exists in the database.

2. Record Update

•**Logic:** Existing postcode records can be updated. The program checks if the record exists before updating.

•**File:** aposlu (Postcode update file)

•**Field:** aposlu_ponr

•**Condition:** The update will only proceed if the record is found in the database.

Configuration and Authorization Rules

1. User Authorization Check

•**Logic:** The program checks user permissions to ensure that only authorized users can make changes to postcode records.

•**Files:**

•l_user (User information)

•w_firm (Firm identifier)

•**Fields:**

•l_user (from l_user)

•w_firm (from w_firm)

•**Condition:** The process will deny access if the user is not authorized.

2. Firm-Specific Data Handling

•**Logic:** The program handles postcode records based on the firm associated with the user.

•**File:** aposl1 (Postcode file)

•**Field:** w_firm
•**Condition:** The program will only process records that match the user's firm identifier.

Financial and Transactional Rules

1. Record Deletion

•**Logic:** Users can delete postcode records. The program prompts for confirmation before deletion.
•**File:** aposlu (Postcode update file)
•**Fields:**
•aposlu_ponr (Postcode number)
•d1ponr (Postcode number for deletion)
•**Condition:** The deletion will only occur if the user confirms the action.

2. Record Display

•**Logic:** The program displays postcode records in a subfile format for user interaction.
•**File:** b1sfl (Subfile for postcode display)
•**Condition:** The subfile will be populated with records based on user selection criteria.

Special Conditions (Program-Specific)

1. Postcode Retrieval (AF010R)

•**Logic:** The program allows retrieval of postcode information based on user input.
•**File:** aposlr (Postcode retrieval file)
•**Field:** aposlr_ponr
•**Condition:** The retrieval will only succeed if the postcode exists in the database. Note: This functionality is critical for ensuring accurate postcode selection.

2. Error Handling for Existing Records (AF010R)

•**Logic:** If a user attempts to create a postcode that already exists, an error message is displayed.
•**File:** aposlr (Postcode retrieval file)
•**Fields:** aposlr_ponr (Postcode number)
•**Condition:** The error message is triggered if the postcode number already exists in the database.

Subprogram Calls Affecting Logic

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

1. dsp_subfile (Display Subfile)

•**Trigger:** Called to display the current state of the subfile.
•**Logic:** This subroutine manages the display of postcode records in the user interface.
•**Impact:** This call acts as a **major logical gateway**, ensuring users see the most up-to-date information.

2. crt_subfile (Create Subfile)

•**Trigger:** Called to populate the subfile with postcode records.
•**Logic:** This subroutine reads records from the database and formats them for display.
•**Impact:** This is a **critical data population step**, ensuring users have access to the correct records.

3. xc2bld (Maintenance Program Call)

- **Trigger:** Invoked when a user wants to create or update a postcode record.
- **Logic:** This subroutine handles the logic for maintaining postcode records, including validation and updates.
- **Impact:** This represents the handoff to the next major business function, ensuring data integrity during updates.

This documentation provides a comprehensive overview of the business logic implemented in the AF010R program, focusing on the maintenance of postcode records within the system.