AP050R.MBR

Path: NXCLOUD/rpgsrc/AP050R.MBR **Generated**: 2026-01-08 15:03:34 **Processing Time**: 14023ms

Business Logic for Profile Manager API Request

This document outlines the business rules that govern the process of interacting with the Profile Manager API, based on an analysis of the RPG program AP050R. The primary focus is on how the program constructs requests to the API and processes the responses.

The core logic for sending a request to the Profile Manager API is contained within the main program logic of AP050R. The program prepares a request file, sends it to the API, and processes the response received from the API.

Request and Response Handling Rules

Profile Manager API Interaction: IFS_Output, IFS_Input

1. Request File Creation

- **Logic:** The program creates a request file that is initially empty and includes a header with the API token.
- **File:** IFS_Output1 (Request file for the API)
- **Field:** w_token
- **Condition:** The request file is created only if the change directory operation to the specified path is successful.

2. Headers File Creation

- **Logic:** A headers file is created to store the API key required for authentication.
- **File:** IFS_Output2 (Headers file for the API)
- **Field:** w_tokeut
- **Condition:** The headers file is created only if the change directory operation to the specified path is successful.

API Call and Response Processing Rules

1. API Call Execution

- **Logic:** The program constructs the necessary parameters and calls the web service to retrieve data from the Profile Manager API.
- **Files:**
- IFS_Output1 (Request file)
- IFS_Input (Response file)
- **Fields:**
- w_url2 (API endpoint URL)
- w_reqt (HTTP method)
- w_reqf (Request file path)
- w_rspf (Response file path)
- **Condition:** The API call is executed only if the request file and headers file are successfully created.

2. Response File Validation

- **Logic:** The program checks if the response file has been created and is accessible after the API call.
- **File:** IFS_Input (Response file)
- **Condition:** If the response file cannot be opened, the API status is set to '9', indicating an error.

Cleanup and Error Handling Rules

1. File Cleanup

- **Logic:** The program attempts to remove temporary files created during the process.
- **Files:**
- IFS_Output1 (Request file)
- IFS_Output2 (Headers file)
- IFS_Input (Response file)
- **Condition:** The cleanup is executed regardless of whether the API call was successful or not.

2. Error Handling

- **Logic:** If an error occurs during the web service call, the program sets the API status to '8' and proceeds to the cleanup routine.
- **Condition:** This error handling routine is triggered upon encountering any failure during the API call.

Subprogram Calls Affecting Logic

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

1. AW702C (Web Service Call)

- **Trigger:** This program is called to execute the web service request.
- **Logic:** It takes parameters such as the URL, request method, request file path, response file path, username, password, encoding, and status.
- **Impact:** This call is critical as it represents the main interaction with the Profile Manager API.

2. AP051R (Response Processing)

- **Trigger:** This program is called to process the response received from the API.
- **Logic:** It takes parameters including the member identifier, response file path, and API status.
- **Impact:** This call is essential for extracting and handling the data returned from the API.

3. AS100R (Initialization)

- **Trigger:** This program is called to initialize parameters and retrieve necessary data before making the API call.
- **Logic:** It retrieves the running number and other parameters needed for the request.
- **Impact:** This call sets up the context for the API interaction, ensuring that all necessary data is available.

This documentation provides a clear overview of the business logic encapsulated in the AP050R program, detailing how it interacts with the Profile Manager API and manages the associated files and error handling.