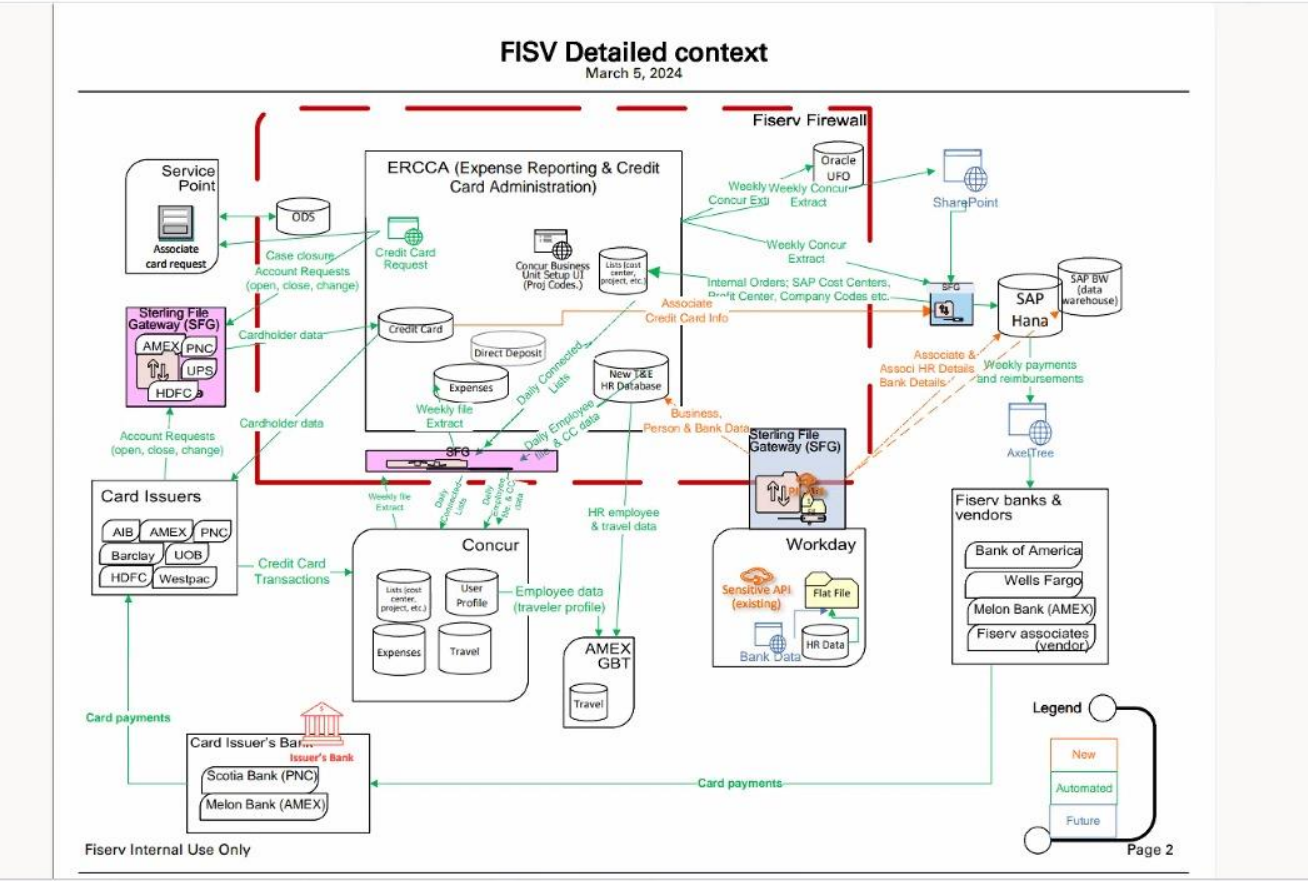
Functional Specification Document:

1. **Introduction:**
   The purpose of this document is to outline the functional specifications for integrating Concur and SAP using middleware to facilitate the automated posting of expense reports submitted by employees.

2. **Integration Architecture:**



3. **Integration Overview:**
   The integration aims to streamline the process of transferring expense report data from Concur, an expense management solution, to SAP, an enterprise resource planning system. Middleware (ERCCA) will serve as the bridge between these two systems, ensuring seamless data exchange.

   The Integration includes the following levels end to end:
   - Receiving the Source data from Concur post the Expenses Reports are finally approved and ready for Accounting Extracts (SAE)
   - Storing the SAE Data in ERCCA (Expense Reporting and Credit Card Administration)
   - Processing the SAE Extract data from Concur and generating the compatible file for SAP to process Accounting entries
   - Scheduling jobs to pick the files to process and Archive the files in Share Point

- **Accounting Extracts (SAE):**
  Input details on Job details of SAE Extract

- **Process in ERCCA:**
  To manage the weekly posting of travel expenses, **Z_Fgli_Weekly_Travel_Cart** will serve as the primary program, delineating the following sub-programs responsible for generating the output files.
    - Z_Fgli_Weekly_Travel_Cart_Top
    - Z_Fgli_Weekly_Travel_Cart_Scr
    - Z_Fgli_Weekly_Travel_Cart_Evnt

  The detailed specifications for each program, along with the corresponding output for every outlined scenario, are provided below.

  Z_Fgli_Weekly_Travel_Cart_Top
  Defines global variables, types, and data structures used throughout the program. This would include definitions for internal tables that store file data, temporary processing data, and any constants required.
    - Collects the SAE Data and stores
    - Identifies the below scenarios based on SAE Extract file and prepare
        - Personal Expenses
        - Billable Expenses
        - Cross Company Allocations
    - Once the scenarios identified, file will be generated with the details of expenses and moves to next program in loop (Z_Fgli_Weekly_Travel_Cart_Scr)

  <Place Holder for User experience screen shot>

  Z_Fgli_Weekly_Travel_Cart_Scr
  Constructs the selection screen for user inputs, such as specifying the file to process or other parameters like processing dates or flags indicating the data source.

  This function aids in generating the document header by extracting all necessary data from the Z_Fgli_Weekly_Travel_Cart_Top program. It sets parameters such as document type, posting date, defaulting debiting expenses to personal expenses, and managing billable expenses.

  Below are the parameters and values defined to create the Header details of Accounting Document:
    - Document Type:
        - If the generated file from the Z_Fgli_Weekly_Travel_Cart_Top is for Personal Expenses Booking, then defaults to ‘ZN’
        - If the generated file from the Z_Fgli_Weekly_Travel_Cart_Top is for Billable Expenses Booking, then defaults to ‘ZM’
    - Posting Date: Automatically defaults to the current system date, corresponding to the day the job is scheduled to execute.
    - General Ledger Account Code:

- If the generated file from the Z_Fgli_Weekly_Travel_Cart_Top is for Personal Expenses Booking, then defaults to ‘120000’
- If the generated file from the Z_Fgli_Weekly_Travel_Cart_Top is for Billable Expenses Booking, then defaults to ‘520500’

<Place Holder for User experience screen shot>

Offsetting Requirements:

Gathers data for offsetting entries associated with the processed travel expenses.

This block ensures that the accounting entries are balanced by specifying the accounts and company codes where offset entries should be posted, thereby ensuring accuracy and completeness in financial records.

- General Ledger Account
- Company Code

<Place Holder for User experience screen shot>

Processing Mode Parameters:

Identifies the origin of the expense report data and captures additional processing preferences. This section is essential for establishing the operational context of data processing, encompassing the data source, and indicating whether the process operates in a test mode.

- **File Source Server:** Users can select the source of the field to be processed. However, the default setting is 'Presentation/Desktop Server,' with the option to switch to 'Application Server' if needed.
- **File Path:** defaulted to C:/, with option to select the exact path of file to be processed
- **File Name:** The filename on the presentation server to be processed. This field's visibility can be dynamically controlled based on previous selections.
- **Email Address:** Optional however, required if the status of the processing files should be sent to specific user to validate
- **Test Mode:** A checkbox allowing the user to indicate if the current run should be a test run, defaulting to checked.

<Place Holder for User experience screen shot>

Z_Fgli_Weekly_Travel_Cart_Evnt

Manages user-triggered events on the selection screen, such as requesting values for specific fields, which may prompt the invocation of search aids or validation logic.

<Place Holder for User experience screen shot>

Z_Fgli_Weekly_Travel_Cart:
This program is designed to automate the weekly import of travel expense details from the ERCCA system (possibly an external travel and expense system like Concur) into SAP. This replaces a manual extract and upload process, streamlining the accounting for travel expenses.

Once the Programs defined below are completed, **Z_Fgli_Weekly_Travel_Cart** will trigger to run the actual postings into Financial system.

- Z_Fgli_Weekly_Travel_Cart_Top
- Z_Fgli_Weekly_Travel_Cart_Scr
- Z_Fgli_Weekly_Travel_Cart_Evnt

<Place Holder for User experience screen shot>

During the processing of the transactions, the program validates the below:

- **Reads Data from Server:** Reads the data from the file uploaded from the Desktop/Application server and executes the accounting postings.
- **Performs Account Postings:**
  i. If there is data in the internal table, the code proceeds to delete any initial rows with empty.
  ii. Performs necessary conversions as required on WBS codes, Cost Centers, Departments, and other conversions.
  iii. Validates Duplicate postings and triggers exception if already processed.
  iv. Validates the Company Codes, General Ledger codes, accounting logics.
  v. Logs all the exceptions and errors if any
  vi. Depending on the selection of Test Mode on initial Screen, runs the BAPI (Business Application Programming Interface) to post Accounting Entries.
  vii. If the mode is selected as Test, displays the logs for validation with numeber of record success and failures along with the error messages for failures.
  viii. If the mode is not selected as Test, then runs the accounting post and logs and trigger an email with the status of the run along with the errors to be handled.

4. Exception Handling:

5.

Z_FGLI_WEEKLY_TRAVEL_CART outlines an extensive business process for handling weekly travel expenses from ERCCA to SAP:

Program Details
- **Program ID**: Z_FGLI_WEEKLY_TRAVEL_CART
- **Title**: Weekly Travel Cart from ERCCA to SAP
- **SAP Area**: FI-GL (Finance-General Ledger)
- **Description**: This program is designed to automate the weekly import of travel expense details from the ERCCA system (possibly an external travel and expense system like Concur) into SAP. This replaces a manual extract and upload process, streamlining the accounting for travel expenses.

Process Flow Overview
1. **Global Data Declaration**:
   - Include File: `z_fgli_weekly_travel_cart_top`
   - Purpose: Defines global variables, types, and data structures used throughout the program. This would include definitions for internal tables that store file data, temporary processing data, and any constants required.

2. **Selection Screen Setup**:
   - Include File: `z_fgli_weekly_travel_cart_scr`
   - Purpose: Constructs the selection screen for user inputs, such as specifying the file to process or other parameters like processing dates or flags indicating the data source.

Functional Specification for Selection Screen: Z_FGLI_WEEKLY_TRAVEL_CART_SCR

Overview
The selection screen `Z_FGLI_WEEKLY_TRAVEL_CART_SCR` is a crucial user interface component of the `Z_FGLI_WEEKLY_TRAVEL_CART` program. It collects user inputs and preferences that dictate how the weekly travel expense data from ERCCA will be processed, validated, and posted to SAP. This screen is divided into multiple blocks, each serving a specific purpose in guiding the user through the setup process for the data processing task.

Detailed Breakdown

Block: Document Type (`doctype`)
- **Purpose**: Allows users to specify document types and posting dates related to the travel expense documents to be processed.
- **Fields**:
  - `p_doctyp`: Specifies the document type for personal expenses, with a default value of 'ZM'.
  - `pdoctyp1`: Specifies the document type for billable expenses, with a default value of 'ZN'.
  - `p_posdat`: Allows the user to set the posting date, defaulting to the current system date (`SY-DATUM`).
  - `p_gl_per`: General Ledger (GL) account number for personal expenses, defaulted to '120000'.
  - `p_gl_bil`: GL account number for billable expenses, defaulted to '520500'.

This block is essential for defining how different types of expenses will be categorized and processed within the SAP system.

Block: Offset (`offset`)
- **Purpose**: Collects information for offsetting entries related to the processed travel expenses.
- **Fields**:
  - `p_gl_off`: GL account number for the offset entry, marked as obligatory.
  - `p_cc_off`: Company code for which the offset entry is to be made, also obligatory.

This block ensures that the accounting entries are balanced by specifying the accounts and company codes where offset entries should be posted.

Block: Processing Mode (`promode`)
- **Purpose**: Determines the source of the expense report data and captures additional processing options.
- **Fields**:
  - `p_pres`: Selection for processing files from the presentation server/desktop, set as the default.
  - `p_appl`: Selection for processing files from the application server.
  - `p_file`: The logical or physical path to the file containing the expense data, with a default suggestion of 'C:\'.
  - `p_fname`: The filename on the presentation server to be processed. This field's visibility can be dynamically controlled based on previous selections.
  - `P_email`: Email address for notifications or processing confirmations. This field can also be shown or hidden dynamically.
  - `p_test`: A checkbox allowing the user to indicate if the current run should be a test run, defaulting to checked.

This block is crucial for defining the operational context of the data processing, including the data source and whether the process is in test mode.

Business Process Interaction
The selection screen serves as the entry point for users to configure the processing of weekly travel expenses. By carefully structuring the input fields and providing sensible defaults, the screen guides the user through setting up the process according to the specific needs of their organizational and accounting practices.

- **Data Sources**: Specifies whether the data will be read from the user's local system (presentation server) or directly from the SAP application server, impacting how the file reading subroutines (`read_file_from_appl` or `read_file_from_pres`) are invoked.
- **Document Processing Parameters**: Influences how the expense reports are categorized and posted, ensuring that each transaction is recorded against the correct document type and GL account.
- **Testing Mode**: Allows users to perform a dry run of the expense processing routine without affecting actual financial records, which is crucial for validation purposes.

3. **Event Handling Setup**:
   - Include File: `z_fgli_weekly_travel_cart_evnt`

- Purpose: Handles events triggered by user actions on the selection screen, including value requests for certain fields, which would invoke search helps or validation logic.

4. **Subroutine: `verify_email`**
   - Triggered By: User requests a value for the email field.
   - Source: Selection Screen event.
   - Purpose: Validates the entered email or provides a search help to select an email. Likely involves checking against a list of valid users or email formats.

Functional Specification for Subroutine: verify_email

Overview
The subroutine `verify_email` is designed to validate and normalize an email address input by the user on the selection screen of the `Z_FGLI_WEEKLY_TRAVEL_CART` program. This routine is critical for ensuring that any email addresses entered for notifications or communications related to the weekly travel expense process are in a valid and standardized format.

Purpose in Main Code
In the context of `Z_FGLI_WEEKLY_TRAVEL_CART`, ensuring the integrity of email addresses is crucial for any subsequent automated communication processes, such as sending confirmations, errors, or reports related to the travel expense processing. This subroutine serves as a validation step before the program proceeds with its main logic.

Detailed Breakdown

Data Types and Structures
- **`sx_addr_type` and `sx_addr`**: Custom types are defined to hold the address type (e.g., SMTP) and the actual address string, aligning with SAP's standard for email address representation.
- **`sx_address`**: A composite structure that includes both the address type and the address itself, encapsulating all necessary details for an email address within the SAP environment.

Constants
- **`cx_t_int`**: Represents the address type for SMTP emails, indicating that the routine is specifically designed to handle internet email addresses.

Data Declarations
- **Local Structures and Variables**:
  - `ls_addr_unst` and `ls_addr`: Instances of the `sx_address` structure for holding unstructured (input) and normalized email addresses, respectively.
  - `lv_address_normal`, `lv_local`, `lv_domain`, and `lv_comment`: Variables for storing the components of the normalized email address and any comments included in the input.

Core Functionality
- **Normalization Process**:

  - The unstructured email address provided by the user (`p_email`) is assigned to `ls_addr_unst`.
  - The function module `SX_INTERNET_ADDRESS_TO_NORMAL` is called with `ls_addr_unst` as input to convert the email address to a normalized format, separating it into local and domain parts while also handling comments.

Exception Handling
- **Error Handling**: The subroutine checks the return code of the function module call. If there's an error (`sy-subrc NE 0`), it triggers a message with the error details. This could indicate issues such as an invalid email format, unsupported address type, or other anomalies detected during normalization.

Business Process Interaction
By verifying and normalizing email addresses at the start of the expense reporting process, `verify_email` ensures that:
- All communication related to the process uses valid email addresses, reducing the risk of failed deliveries.
- Email addresses are stored and used in a consistent format, facilitating better data management and integrity.
- The system can reliably notify users about the status of their expense report processing, including any manual interventions required due to errors or exceptions encountered.

Data Sources and Destinations
- **Source**: The user's input on the selection screen (`p_email`).
- **Destination**: Processed email addresses are potentially used within the program for notifications or logging but are not directly saved or exported by this subroutine.

5. **Subroutine: `f4_file_name`**
   - Triggered By: User requests a file name through the F4 help.
   - Source: Selection Screen event.
   - Purpose: Offers a file selection dialog to the user, facilitating the choice of a file from a predefined directory or list.
Functional Specification for Subroutine: F4_FILE_NAME

Overview
The subroutine `F4_FILE_NAME` is designed to facilitate the selection of a file name via an F4 help request. Depending on the user's previous selection between processing data from the application server (`P_APPL`) or the presentation server (`P_PRES`), it dynamically calls another subroutine to provide the appropriate file selection dialog. This adaptability enhances user experience by offering a context-sensitive file selection mechanism.

Purpose in Main Code
Within the larger program `Z_FGLI_WEEKLY_TRAVEL_CART`, this subroutine plays a critical role in ensuring that the user can efficiently and correctly specify the source file for the weekly travel expense data. This file contains vital information needed for the subsequent processing, validation, and posting routines within the SAP system.

Detailed Breakdown

1. **Data Preparation and Dynamic Read**:

- **Internal Table `i_fieldvalues` Usage**: This table is used to temporarily store values read from the screen fields. It is particularly focused on capturing whether the user intends to process a file from the application server (`P_APPL`) or the presentation server (`P_PRES`).
- **Function Module `DYNP_VALUES_READ`**: This SAP standard function module reads the current values of the screen fields. It is used here to determine the source of the file selection based on user input.
  - **Export Parameters**:
    - `dyname`: Contains the report/program name (`sy-repid`), ensuring the function reads values from the current program's selection screen.
    - `dynumb`: Uses the current screen number (`sy-dynnr`) to precisely locate the field values.
  - **Table Parameter**:
    - `dynpfields`: Passes `i_fieldvalues` to fetch the values of specific dynamic program fields.
  - **Exception Handling**: Multiple exceptions are handled to catch various errors ranging from invalid field names to more complex issues like double conversion errors or undefined errors.

2. **Decision Logic for File Source**:
- After reading the values, the subroutine checks if the `P_APPL` field is marked (`'X'`). This determination influences which file selection dialog subroutine is called next.
  - **`P_APPL` Marked**: Indicates that the user wishes to select a file from the application server. The `get_f4_application` subroutine is called in this case.
  - **Otherwise**: The user intends to select a file from the local system (presentation server), triggering the `get_f4_presentation` subroutine.

3. **Subroutines Called**:
- **`get_f4_application`**: This routine provides a dialog for the user to select a file from the application server.
- **`get_f4_presentation`**: Similar in purpose to `get_f4_application`, but focuses on the user's local system.

Business Process Interaction
This subroutine is an integral part of the data collection phase of the weekly travel expense processing workflow. By allowing users to specify the file source accurately, it ensures that the correct data set is selected for processing. This step is critical for maintaining data integrity and accuracy in financial postings related to travel expenses.

Data Tables and Sources
- **`i_fieldvalues`**: An internal table used to store and read dynamic screen values to determine the user's choice of file source.
- No direct interaction with persistent data tables is performed in this subroutine. However, it plays a vital role in setting up the correct pathway for data ingestion into the system.

6. **Subroutine: `change_screen_fields`**
- Triggered By: Selection Screen output.
- Purpose: Dynamically adjusts the selection screen fields based on certain conditions, like hiding or showing fields, setting default values, etc.

Functional Specification for Subroutine: change_screen_fields

Overview
The `change_screen_fields` subroutine dynamically adjusts the input and visibility of fields on the selection screen based on the user's selections within the `Z_FGLI_WEEKLY_TRAVEL_CART` program. This customization enhances user experience by streamlining the input process, ensuring only relevant options are available based on previous selections, particularly regarding the source of the expense report data.

Purpose in Main Code
In the `Z_FGLI_WEEKLY_TRAVEL_CART` program, this subroutine supports the primary goal of configuring the data processing environment tailored to the user's needs. It helps prevent user errors and simplifies the data entry process by dynamically modifying the selection screen based on the context of the process being executed.

Detailed Breakdown

Data Retrieval from Custom Table
- **Custom Table**: `zggd_gen_xref`
  - This table is queried to retrieve specific GL account numbers (`p_gl_off`) and company codes (`p_cc_off`) that are predefined for the ERCCA interface (`identifier = 'ERCCA_W'`). The selection criteria include an identifier for the ERCCA Weekly process and an active status flag.

Dynamic Screen Modifications
- **Presentation Server Selection (`p_pres`)**:
  - If the user opts to process files from the presentation server, certain fields grouped under `M1` are made non-inputtable and invisible, streamlining the interface for this data source choice. The default file path is set to 'C:\' unless already specified.

- **Application Server Selection (`p_appl`)**:
  - Conversely, when the application server is selected, fields related to this choice are enabled, and the default logical file path might be updated based on specific interface configurations, like `Z_RTR_IN_CONCUR_OFD_WEEKLY` for OFD interfaces.

Conditional Field Visibility and Input State
- The subroutine loops through all screen elements, adjusting the input capability and visibility based on the user's data source selection (`p_pres` or `p_appl`) and groups them into `M1`. This ensures that only relevant fields are interactable, depending on whether the data will be loaded from the application server or the presentation server.

- Additionally, the GL account and company code fields for offset (`p_gl_off` and `p_cc_off`) are set to non-inputtable, indicating these values are predetermined and not subject to change during this process.

Business Process Interaction
This subroutine directly interacts with the business process of importing and processing weekly travel expenses by:
- Ensuring users are only presented with relevant fields, reducing complexity and potential confusion.

- Automatically populating certain fields based on the program's configuration, which aids in standardizing the data entry process and ensuring consistency across expense reporting.

Data Sources and Destinations
- **Source**: User selections on the selection screen and predefined configurations stored in the `zggd_gen_xref` table.
- **Destination**: User interface (selection screen) of the `Z_FGLI_WEEKLY_TRAVEL_CART` program, where the visibility and input state of fields are dynamically adjusted.

7. **Data Processing Initiates**:
   - Phase: Start of Selection.
   - Description: The core logic begins, including reading the file specified by the user, processing the data, and preparing it for posting in SAP.

8. **Subroutine: `read_file_from_appl`**
   - Source: Application Server file specified by the user.
   - Purpose: Reads data from a file stored in the application server. This involves opening the file, reading its contents line by line, and storing the data in an internal table for further processing.

Functional Specification for Subroutine: READ_FILE_FROM_APPL

Overview
The `READ_FILE_FROM_APPL` subroutine is tasked with reading a file from the application server within the SAP environment. This routine is a critical component of the `Z_FGLI_WEEKLY_TRAVEL_CART` program, specifically designed to handle the ingestion of weekly travel expense data stored on the SAP application server.

Purpose in Main Code
This subroutine supports the main functionality of the `Z_FGLI_WEEKLY_TRAVEL_CART` program by importing travel expense report data for processing. It ensures that data from the application server can be accessed, read, and subsequently processed for expense report generation or update within the SAP system.

Detailed Breakdown

Data Preparation
- **Variable `v_file2`**: Formed by concatenating `file_name` (likely provided or determined earlier in the program to specify a directory or path) with `p_fname` (the filename input by the user on the selection screen). This operation ensures the program dynamically targets the correct file as specified by the user.

Opening the Dataset
- **Operation**: `OPEN DATASET` statement is used to open the file on the application server for reading. The mode is specified as text (`IN TEXT MODE`), and the default system encoding is used (`ENCODING DEFAULT`).
- **Error Handling**: Checks the return code (`sy-subrc`). If the operation fails (`sy-subrc NE 0`), an error message is displayed, and the subroutine exits, preventing further processing of an unavailable dataset.

Reading the File
- **Loop for Reading**: A `DO` loop is initiated to read each line of the dataset into the work area `wa_appl` using the `READ DATASET` statement.
  - **Exit Condition**: The loop exits if `READ DATASET` sets `sy-subrc` to a non-zero value, indicating the end of the file or a read error.

- **Record Processing**: For each line successfully read into `wa_appl`, the subroutine `split_record` is called. This suggests that `split_record` is responsible for parsing and possibly validating each line of the file according to the expected format of the expense data.

Cleanup
- **Initial Record Deletion**: The statement `DELETE i_file INDEX 1` is executed after reading the file, indicating the removal of the first line in the `i_file` internal table. This could imply that the first line of the file contains headers or other non-data content that should not be processed further.

Business Process Interaction
- Ensures data integrity by programmatically verifying file accessibility and readability, essential for automated processing workflows.
- Facilitates accurate expense report generation by ensuring only valid, well-formed data is processed, as indicated by the call to `split_record` for each line read.

Data Sources and Destinations
- **Source**: Application server file system, specifically the file path and name specified by the user or determined by prior program logic.
- **Destination**: Internal processing logic of `Z_FGLI_WEEKLY_TRAVEL_CART`, with data lines being parsed by `split_record` and potentially populating internal tables for further processing.

9. **Subroutine: `read_file_from_pres`**
   - Source: Desktop (Presentation Server).
   - Purpose: Similar to `read_file_from_appl`, but for files selected from the user's desktop. This allows for a more direct upload process for users.

Functional Specification for Subroutine: READ_FILE_FROM_PRES

Overview
The `READ_FILE_FROM_PRES` subroutine is designed to facilitate the reading of travel expense data from a file located on the presentation server (typically the user's local machine). This routine handles the import of data, and prepares it for processing within the SAP environment. This process includes validation checks and data structure preparation to align with the program's processing requirements.

Purpose in Main Code

In the context of the `Z_FGLI_WEEKLY_TRAVEL_CART` program, this subroutine directly supports the initial data collection phase for processing weekly travel expenses. By enabling users to upload data from their local systems, it broadens the accessibility of the program and facilitates the inclusion of data that may not be directly accessible from the application server.

Detailed Breakdown

Initial Setup and Data Preparation
- **Variables Initialization**: Refreshes the `i_file` internal table, which is used to store the processed data from the file.
- **Determination of File Name**: The file name (`v_filename`) is prepared based on user input from the selection screen, potentially concatenating the file path and file name for precise identification.

File Reading Process
- **Function Module for File Conversion**:
  - **`ZKCD_CSV_FILE_TO_INTERN_CONVRT`**: A custom function module that reads a CSV file and converts its contents into an internal table format suitable for further processing. This function handles the specifics of CSV formatting, including the separator character.
  - **Parameters**: Includes the file name and the separator character, indicating a tailored approach to CSV file handling.
  - **Exception Handling**: Specific exceptions related to file upload and format are handled, ensuring that any issues encountered during the file reading process are appropriately addressed.

Data Validation and Flag Setting
- **Original First Data Employee Check**: A specific check is performed on the data to identify records associated with "Original First Data" employees. This involves searching for a predefined value in a specific row and column of the file, setting a flag (`v_ofd_flag`) based on the result. This flag presumably influences subsequent processing logic tailored to these employees.

Data Collection and Transformation
- **Data Extraction**: Loops through the converted internal table (`ta_kcde_cells`), extracting data and assigning it to fields within a work area (`wa_file`). This step also includes specific transformations, such as converting certain fields to upper case.
- **Row and Header Management**: Special handling for the first row of the file, potentially used for headers, ensures that header data is processed separately or flagged accordingly.

Finalization and Cleanup
- **Data Appending**: Processed records are appended to the `i_file` internal table, preparing them for the next stages of the program's processing logic.
- **Deletion of First Index**: The first index of the `i_file` internal table is deleted, which might be a header or an otherwise unneeded row, ensuring the data set is clean for processing.

Business Process Interaction

This subroutine plays a crucial role in importing and initial processing of data for the `Z_FGLI_WEEKLY_TRAVEL_CART` program, enabling users to integrate local data files into the SAP processing environment. It specifically caters to scenarios where the data source is the presentation server, offering flexibility in data collection methods for the program.

Data Sources and Destinations
- **Source**: User-specified file from the presentation server, containing travel expense data.
- **Destination**: The `i_file` internal table within SAP, where the data is stored for further processing by the program.

10. **Subroutine: `fetch_wbs_suffix`**
    - Purpose: Fetches WBS (Work Breakdown Structure) suffix information, which is likely used to append or modify project codes or expense categorization within the data being processed.

Functional Specification for Subroutine: FETCH_WBS_SUFFIX

Overview
The `FETCH_WBS_SUFFIX` subroutine is designed to retrieve Work Breakdown Structure (WBS) suffixes associated with different types of expenses from a custom SAP table, `zflgi_exp_wbs`.

Purpose in Main Code
Within the `Z_FGLI_WEEKLY_TRAVEL_CART` program, this subroutine underpins a foundational aspect of the data processing workflow—ensuring that expense data is correctly associated with the relevant WBS elements.

Detailed Breakdown

Data Retrieval
- **Database Table**: `zflgi_exp_wbs`
  - A custom table that stores mappings between expense types (`exp_type`) and their respective WBS suffixes (`suffix`). This table acts as a reference for determining how different expenses should be allocated within the SAP system.

- **Selection Operation**:
  - A `SELECT` query is executed without specific key criteria, implying the intention to fetch all records from `zflgi_exp_wbs`. This comprehensive data retrieval suggests that the program may later filter or lookup specific records dynamically based on the expense types encountered in the input data.

Target Data Structure
- **Internal Table `i_wbs`**:
  - Serves as the destination for the fetched records. This internal table is structured to hold the `exp_type` and `suffix` fields, mirroring the structure of the source database table.
  - By loading all relevant WBS suffix mappings into memory, the program can efficiently access and utilize these mappings throughout the expense processing workflow.

Business Process Interaction
The retrieval and application of WBS suffixes directly influence how the program categorizes and reports expense data within the broader SAP environment. For organizations utilizing SAP's project system or similar modules, the accurate association of expenses with WBS elements is essential for:
- Tracking project costs and budget adherence.
- Facilitating detailed project or cost center reporting.
- Ensuring compliance with internal and external accounting standards.

Data Sources and Destinations
- **Source**: The custom SAP table `zflgi_exp_wbs`, which is queried to obtain the full set of expense type-to-WBS suffix mappings.
- **Destination**: The `i_wbs` internal table, which stores these mappings within the program's runtime environment for immediate reference during data processing.

11. **Data Enrichment and Validation**:
    - Description: The program processes the raw data from the file, applying business logic to categorize expenses, assign them to the correct accounts, and validate the entries for correctness.

12. **Subroutine: `fetch_bkpf`, `fetch_pernr`, `check_postings`, `fetch_company_codes`**
    - Purpose: These routines likely fetch additional data required for postings, such as validating existing postings, fetching personnel numbers, company codes, and preparing document headers (BKPF) for financial postings.

13. **Posting and Error Handling**:
    - Description: After validation and preparation, the program posts the expenses to SAP FI. It handles any errors encountered during the posting process and logs these for review.

14. **Subroutine: `call_posting_bapi`**
    - Purpose: Calls the BAPI to post the financial documents to SAP based on the prepared data. This subroutine handles the construction of the BAPI call, passing the necessary data, and interpreting the response.

15. **Subroutine: `display_log` and `send_mail`**
    - Purpose: `display_log` displays a log of the processing results, including errors and successful postings. `send_mail` may send a notification email summarizing the process outcomes.

16. **Finalization**:
    - User interaction is managed through PF-STATUS and USER-COMMAND handling

, guiding the user through any final steps or corrections.

Key Data Tables and Sources
- **filenameci**, **filetextci**: Custom tables used in file selection logic, storing file metadata and descriptions.

- **Internal Tables**: Various internal tables like `i_logfi`, `i_file`, `i_repid`, etc., are used for processing and temporary data storage.