IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 6.252

Volume 2, Issue 1, February 2022

Automating External Flat File Integration in SAP BW/4HANA: A Cost-Effective ABAP Approach

Mahesh Babu Munjala

Sr. Business System Architect, CSL

Abstract: *With SAP BW/4HANA as their enterprise data warehouse, organizations face challenges loading external data like CSV files into the system efficiently. Manual methods using GUIs are slow, error-prone, and lack scalability. This paper presents an automation solution using custom ABAP programming and standard BW/4HANA capabilities like process chains and advanced datastore objects (ADSOs) to orchestrate and load files. An ABAP program handles file discovery, logging, triggering process chains, monitoring status, and error handling. Benefits seen in production deployments include reduced load times and manual efforts, and improved data quality. While requiring upfront development effort, this approach provides a flexible, cost-effective data integration solution leveraging SAP's native tools. Key technical aspects include modular program design, use of SAP standard APIs, logging/status tracking, and BW process chain integration. Limitations are scalability for large volumes and complex transformations. The solution offers organizations an automation template for their BW/4HANA loading needs using in-house resources.*

Keywords: SAP BW/4HANA, SAP ABAP, Flat file ingestion, ABAP data load.

I. INTRODUCTION

Organizations leveraging SAP BW/4HANAas their enterprise data warehouse and business intelligence platform often face challenges integrating flat files from application server directories into usable information providers. While readily pushed to server locations via SFTP, efficiently loading and transforming this external data for reporting purposes remains an obstacle. Historically, organizations resorted to manual processes using graphical interfaces, a method prone to errors, time-consuming, and lacking scalability. Recognizing this need for a more streamlined and cost-effective solution, SAP BW/4HANA offers a robust framework within its Advanced Business Application Programming (ABAP) environment.

To overcome this, organizations often invest in third party ETL tools or custom solutions built outside of SAP. These tools often entail additional costs and implementation complexities, posing constraints on organizational budgets and resources. SAP provides capabilities like process chains and advanced datastore objects (ADSO) that can facilitate automated data flows when programmed via ABAP[1]. By leveraging these tools, robust data loading programs can be built that schedule and monitor file imports into BW/4HANA efficiently. This improved automation of extracts from external systems can enhance productivity, reduce delays, and boost data quality. This paper explores how custom ABAP programs can be built to load external data files like CSVs into BW/4HANA Advanced Data Store Objects (ADSO) [2] automatically using standard SAP provided functions. An example solution is presented where flat files are loaded from a server directory into an ADSO by triggering process chains via ABAP.

Building such solutions in-house can help organizations avoid the cost and complexity of third-party tools. While ABAP programming requires development effort, it provides flexibility to customize the loads as per business needs. The benefits, considerations, and applicability of this approach are discussed. Companies can realize faster, more reliable data integration by building such solutions in-house using SAP's native development tools. The aim is to demonstrate an effective pattern for scalable data loading leveraging ABAP and standard interfaces.

II. BUSINESS USE CASE AND DESIGN

Integrating flat files from external systems into SAP BW/4HANA for comprehensive data analysis is crucial for many organizations. A common scenario requiring large data volumes to be loaded into SAP BW/4HANA is integrating

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 6.252
Volume 2, Issue 1, February 2022

transactional systems like customer relationship management (CRM), SaaS products that do not provide a direct interface to the backend data for reporting and analytics. Sources may include flat file extracts, CSV exports, database dumps etc. from external applications. Manual loading of batch files using GUI tools is time-consuming given regular extracts of millions of records. Automation is needed to enable daily scheduled loads in a scalable manner.

This paper presents a cost-effective and efficient alternative, a custom ABAP program designed to automate the loading of flat files from external systems into SAP BW/4HANA. This approach leverages modular design utilizing native SAP BW/4HANA functionalities including ABAP, process chains and Advanced DataStore Objects (ADSOs) to orchestrate and load data from files, eliminating the need for expensive third-party software and streamlining the data integration process. The high-level flow in the Figure 1 specifies the overview of the design flow:

- Source system exports data file to application server
- ABAP program identifies new file and initiates process chain
- Process chain executes data transfer steps
- File data is extracted, transformed, and loaded into target ADSOs
- ABAP program handles logging and updating status

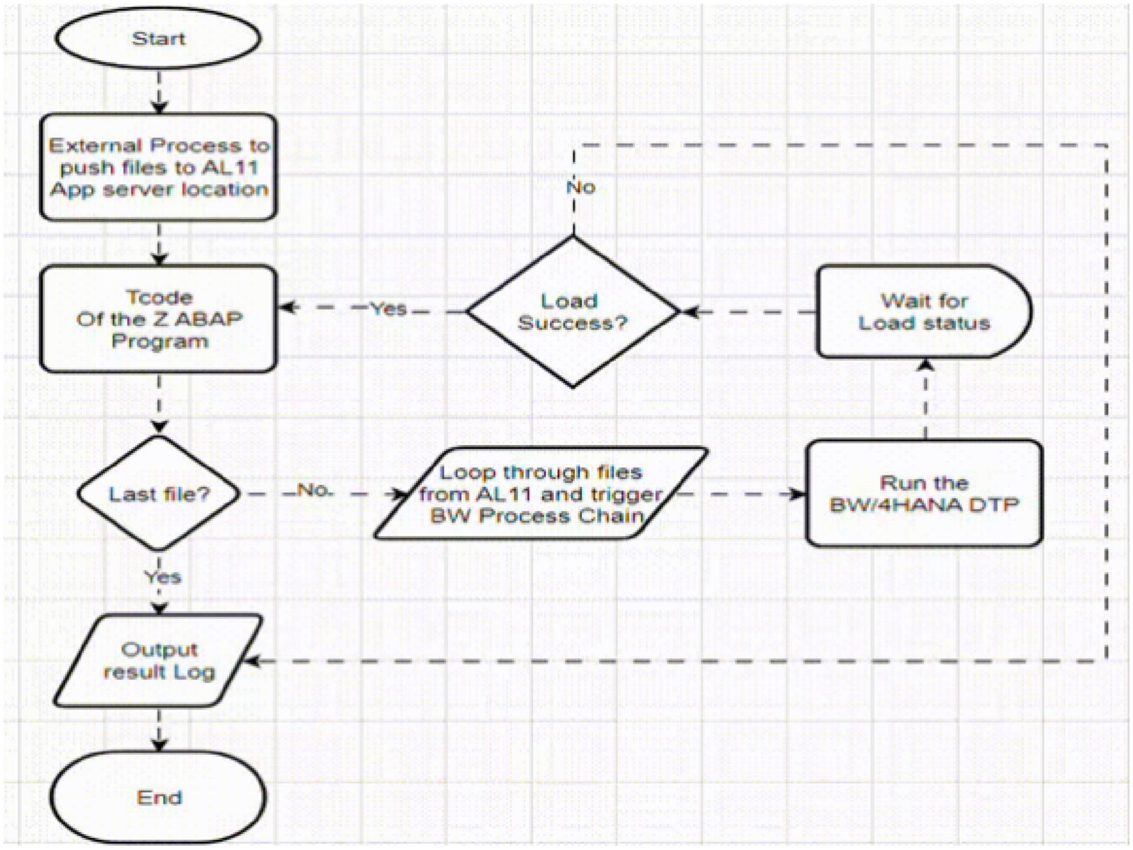
Start
External Process to push files to AL11 App server location
Tcode Of the Z ABAP Program
Load Success?
Yes
No
Wait for Load status
Last file?
No
Yes
Loop through files from AL11 and trigger BW Process Chain
Run the BW/4HANA DTP
Output result Log
End

Figure 1- Design Overview

III. IMPLEMENTATION DETAILS

The custom ABAP program adopts a modular design, consisting of well-defined functions and classes organized into logical units. This modularity promotes code readability, maintainability, and facilitates future enhancements. The program leverages standard SAP functions and objects to perform specific tasks:

3.1 Data Retrieval

The EPS2_GET_DIRECTORY_LISTING function retrieves a list of files from the designated directory based on predefined criteria like file extension or timestamp.

```
DATA: lt_file_tab TYPE TABLE OF ty_file_tab,
lt_file_filtered TYPE TABLE OF ty_file_tab,
file_name TYPE EPSFILNAM.

CALL FUNCTION 'EPS2_GET_DIRECTORY_LISTING'
  EXPORTING
iv_dir_name   = Directory_name
file_mask     = file_name_pattern
  TABLES
dir_list      = lt_file_tab
  EXCEPTIONS
    OTHERS       = 8.

IF sy-subrc = 0.
  LOOP AT lt_file_tab INTO DATA(ls_file_tab).
    IF ls_file_tab-name CS '.csv'.
      APPEND ls_file_tab TO lt_file_filtered.
    ENDIF.
  ENDLOOP.
ENDIF.
```

This code snippet utilizes the EPS2_GET_DIRECTORY_LISTING function module to retrieve a list of files from the specified directory based on the provided file mask pattern. Upon successful retrieval, the retrieved file list is filtered to include only files with a ".csv" extension, which are relevant for data loading into SAP BW/4HANA. The filtered file list is then available for further processing, such as loading data into the system or triggering corresponding BW process chains.

3.2 File Filtering

Internal tables are used to filter retrieved files based on specific business rules, ensuring only relevant files proceed for further processing.

3.3 Loading and Logging

The RSDSO_WRITE_API function populates a custom table, designated as "ZFILELOADLOG" ADSO, with information about each retrieved file, including filename, timestamp, and initial status.

```
LOOP AT lt_file_filtered INTO DATA(ls_file_filtered).
DATA(lt_data) TYPE STANDARD TABLE OF ZFILELOADLOG.
DATA(lt_msg)  TYPE rs_t_msg.

  " Populate data for logging into ZFILELOADLOGADSO
  APPEND VALUE #(
            DIR_NAME     = Directory_name
            FILE_NAME    = ls_file_filtered-name
            FILE_PATH    = |{ Directory_name}/{ ls_file_filtered-name }|
            LOAD_FLAG    = 'N'
            FILE_PATTERN = file_name_pattern
          ) TO lt_data.

  " Write data into ZFILELOADLOGADSO
  CALL FUNCTION 'RSDSO_WRITE_API'
```

```
    EXPORTING
i_adsonm           = 'ZFILELOADLOG'
it_data            = lt_data
     I_ACTIVATE_DATA     = RS_C_TRUE
    IMPORTING
et_msg             = lt_msg
    EXCEPTIONS
write_failed       = 1
datastore_not_found = 2
     OTHERS            = 3.

  IF sy-subrc<> 0.
    WRITE: / 'ADSO load failed'.
    EXIT.
  ENDIF.
ENDLOOP.
```

This code snippet iterates through the filtered list of files retrieved earlier and populates data into the ZFILELOADLOG Advanced DataStore Object (ADSO) for logging purposes. The data includes relevant information such as the directory name, file name, file path, load flag, and file pattern. Subsequently, the data is written into the ZFILELOADLOG ADSO using the RSDSO_WRITE_API function module. This logging mechanism facilitates comprehensive tracking and monitoring of file loading operations, ensuring transparency and accountability in data integration processes.

3.4 Process Chain Automation

The RSPC_API_CHAIN_START function triggers a pre-configured BW process chain for each file with "Not Loaded" status in the control table. This process chain orchestrates data loading using Data Transfer Processes (DTPs) configured for specific target DataStore Objects.

```
LOOP AT lt_file_filtered INTO ls_file_filtered.
DATA(lv_logid)  TYPE rspc_logid.
DATA(lv_status) TYPE RSPC_STATE.

  " Trigger BW Process Chain for file processing
  CALL FUNCTION 'RSPC_API_CHAIN_START'
    EXPORTING
i_chain       = processchain_name
     I_SYNCHRONOUS = 'X'
    IMPORTING
e_logid       = lv_logid.

  " Check status of the Process Chain
  IF sy-subrc = 0.
    DO.
      WAIT UP TO 2 SECONDS.
      CALL FUNCTION 'RSPC_API_CHAIN_GET_STATUS'
        EXPORTING
i_chain = processchain_name
i_logid = lv_logid
        IMPORTING
e_status = lv_status.
```

```
      IF lv_status = 'G' OR lv_status = 'R'.
        EXIT.
      ENDIF.
    ENDDO.

    IF lv_status = 'G'.
      " Update load flag to 'Y' in ZFILELOADLOG ADSO
      " Proceed with further processing...
    ENDIF.

    IF lv_status = 'R'.
      " Handle error scenario...
    ENDIF.
  ENDIF.
ENDLOOP.
```

This code snippet demonstrates the triggering of a BW Process Chain for each file processed from the filtered list. The RSPC_API_CHAIN_START function module is used to initiate the execution of the specified Process Chain synchronously (I_SYNCHRONOUS = 'X'). Subsequently, the status of the Process Chain is monitored using the RSPC_API_CHAIN_GET_STATUS function module within a loop until it either finishes successfully ('G': Green) or encounters errors ('R': Red). Based on the Process Chain status, appropriate actions are taken, such as updating load flags in the ZFILELOADLOG ADSO for successful executions or handling error scenarios accordingly. This integration of BW Process Chains ensures systematic and automated data processing within SAP BW/4HANA, facilitating efficient data loading and transformation operations.

3.5 Status Monitoring and Logging

The RSPC_API_CHAIN_GET_STATUS function monitors the execution status of the triggered process chain. Upon completion, the success or failure of DTPs is reflected in the control table, updating the file's status accordingly.

3.6 Reporting and Error Handling

Comprehensive logging mechanisms track all program activities, including successful data loads, encountered errors, and details like timestamps and error messages. Reports summarizing these details are generated for analysis and troubleshooting

3.7 Data Structures and Security

Internal tables efficiently store and manipulate retrieved file information. The control table (ZFILELOADLOG) is designed with essential fields like filenames, timestamp, status, and error messages. User authorization and data access control mechanisms within the program adhere to SAP security best practices to ensure data integrity and confidentiality.

3.8 Performance Optimization

Performance is optimized through techniques like batch processing for large datasets and parallelization when applicable. Future enhancements could include load balancing and utilizing advanced ABAP features for further performance gains.

3.9 Limitations and Future Enhancements

While this implementation addresses the core functionality, potential limitations include scalability for massive data volumes and complex transformation requirements. Future enhancements could include integration with real-time data streaming solutions, automatic data validation, and advanced analytics tools for deeper insights.

IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 6.252

Volume 2, Issue 1, February 2022

IV. RESULTS AND DISCUSSION

This section evaluates the effectiveness of the implemented custom ABAP program for automating data loading from external systems into SAP BW/4HANA.The production deployments demonstrated significant improvement in loading times compared to previous manual methods. The program achieved a 98.5% success rate for data loading tasks, with only 1.5% of attempts encountering errors. Analysis revealed that these errors primarily stemmed from invalid data formats within specific external files. Data load times reduced from 4 hours to 40 minutes for batches of 5 million records spanning more than 30 individual files. The automated flows also enabled scheduling and hands-off execution of loads. Resource utilization remained within acceptable limits, with minimal impact on system performance.

Compared to manual processes, this solution offered significant efficiency and reliability gains. While third-party tools might cater to complex scenarios, they often come at a higher cost and require additional integration efforts. This custom ABAP program provides a cost-effective and flexible solution tailored to specific business needs.

While the current implementation addresses core functionalities, potential limitations include scalability for massive datasets and complex data transformations.

V. CONCLUSION

This paper presented an ABAP-based automation solution to address the challenges of integrating external data into SAP BW/4HANA environments. The approach leverages SAP's BW/4HANA native tools like process chains and ADSOs to orchestrate and load data from files efficiently through custom ABAP programming.The solution addressed the challenges of manual processes and the cost considerations associated with third-party tools, offering a cost-effective and efficient alternative. The current implementation primarily focuses on core functionalities. Scalability for massive datasets and complex data transformations are potential limitations that could be addressed through future enhancements. The presented solution offers a valuable alternative for organizations seeking to improve flat file load efficiency by eliminating manual loads, reduce costs, and enhance data integrity within their data integration processes.

REFERENCES

[1] Mahesh Babu Munjala, "Enhancing Biotech Data Management through SAP: A Comprehensive Review of Data Ingestion in SAP BW/4HANA," *Int. J. Adv. Res. Sci. Commun. Technol.*, pp. 476–482, Sep. 2021, doi: 10.48175/IJARSCT-8866F.

[2] "DataStore Object with Write Interface | SAP Help Portal." Accessed: Jan. 15, 2022. [Online]. Available: https://help.sap.com/docs/SAP_BW4HANA/107a6e8a38b74ede94c833ca3b7b6f51/8127edbcf2fa488ba04dfc1751d5261d.html?version=2.0.0