How to get data from a submit program in SAP ABAP

This guide shows you how to get data from a submit program in SAP ABAP when the program displays an ALV (ABAP List Viewer) report. You first confirm that the program is a report and identify the structure or transparent table behind the ALV output. Then you write a program that submits the report, captures its output in memory, and loads it into an internal table. Finally, you check the result in the ABAP Debugger.

Prerequisites

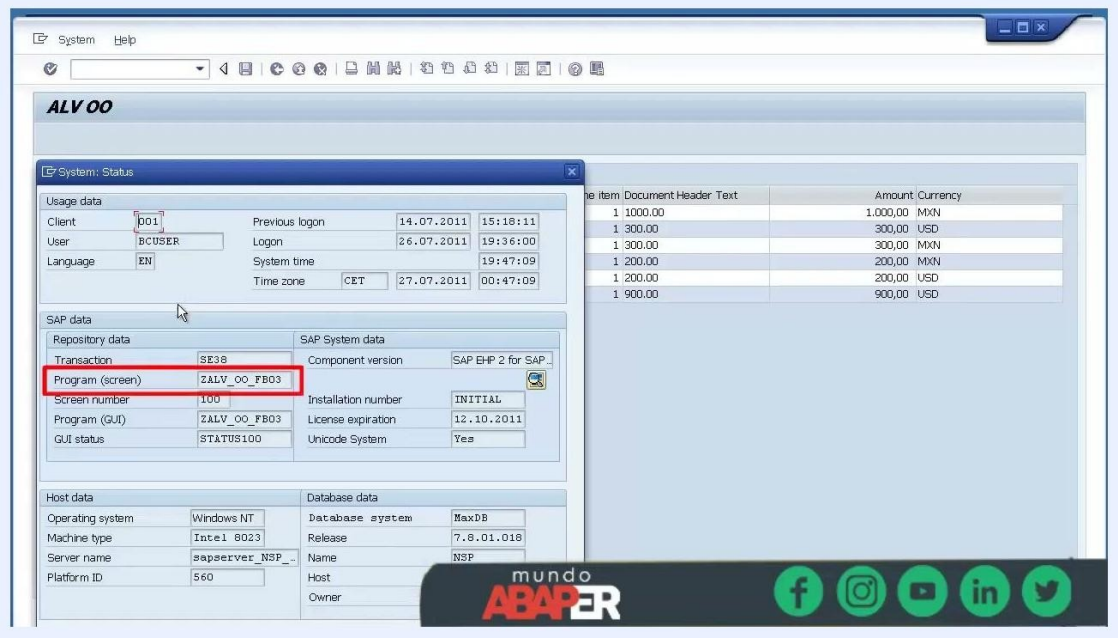
Before you start, make sure you have the following:

- A transaction that displays the ALV report you want to read. This guide uses an ALV report screen titled "ALV OO".
- An associated program of type **Report**. You can use the SUBMIT statement only if the program behind your transaction is a report.
- Access to the Data Dictionary, in case you need to create a custom Z structure.
- Access to the ABAP Editor and the ABAP Debugger.

Step 1: Run the ALV report

1. Open the transaction that displays the ALV report.
2. On the selection screen, enter P_BUKRS = 1000 and P_GJAHR = 2011.

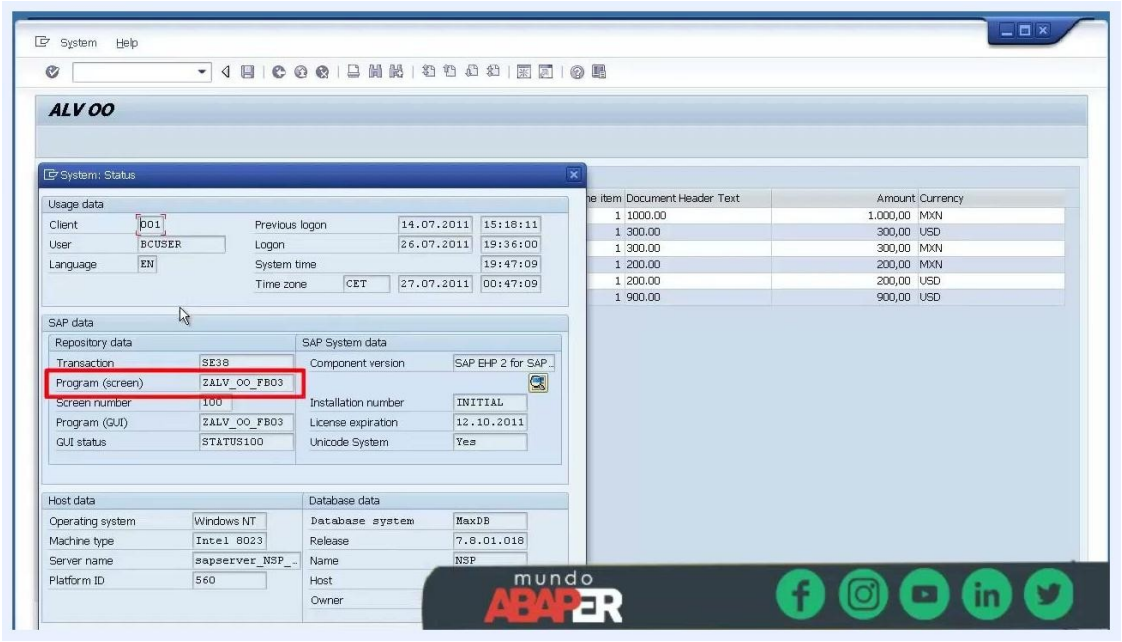
3. Click **Execute** (or press F8) to run the report.



ALV report selection screen with fields P_BUKRS = 1000 and P_GJAHR = 2011, and the Execute button being clicked

Step 2: Open the system status

With the ALV report displayed, open the top menu and choose **System** > **Status...**.



SAP GUI with the ALV report displayed, the System menu open, and the Status option highlighted

Step 3: Verify the program type

1. In the **System: Status** window, find the **Program (screen)** field and note the program name, for example ZALV_OO_FB03.
2. Double-click the program name to open its properties.
3. Go to the **Attributes** section.
4. Confirm that **Type** is set to **Report**.

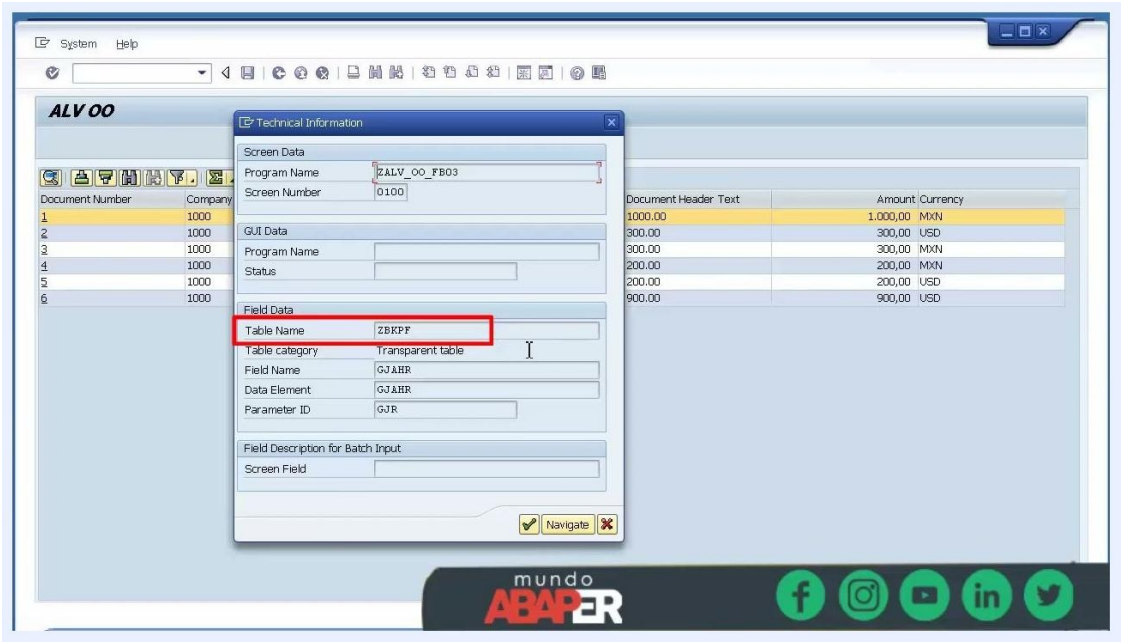
If the program is not a report, you cannot use SUBMIT to get its data.

Step 4: Identify the structure or transparent table used in the ALV

1. Click any cell in the ALV grid.
2. Press F1 to open the field help.
3. In the **Technical Information** window, note the **Table Name** (for example, ZBKPF), the **Table Category** (for example, Transparent

table), the **Field Name** (for example, GJAHR), and the **Data Element** (for example, GJAHR).

You need the table or structure name later to process the report data after the submit.



Technical Information window showing Table Name ZBKPF, Table Category Transparent table, and other field details

Different ALV reports often use different structures or transparent tables. Always confirm the correct structure or table for each report you want to process with SUBMIT.

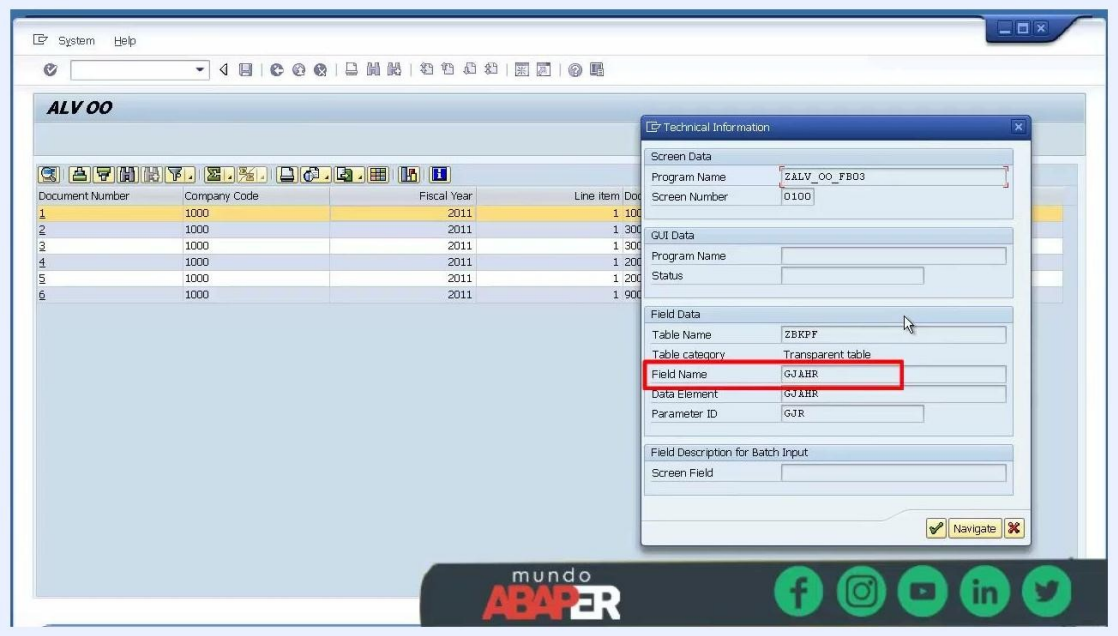
Step 5: Create a Z structure in the Data Dictionary (if needed)

If the ALV report fields come from different structures (for example, BSEC and T001), create a custom Z structure in the Data Dictionary. Include all the fields you need from the ALV output, or only the ones your processing requires.

Make sure each field in your Z structure:

- Has the same name as in the original structure or table.
- Uses the same data element as in the original definition.

If you use different field names or data types, the SUBMIT statement does not recognize the fields. Those fields stay empty in your internal table.



Technical Information window showing Table Name ZBKPF, Field Name GJAHR, and Data Element GJAHR, which your Z structure fields must match

Step 6: Create a new program to submit to the ALV report

Create a new ABAP report program, for example Z_SUBMIT_2. This guide puts all the code in one program for clarity. In practice, modularize your code with includes and a proper structure.

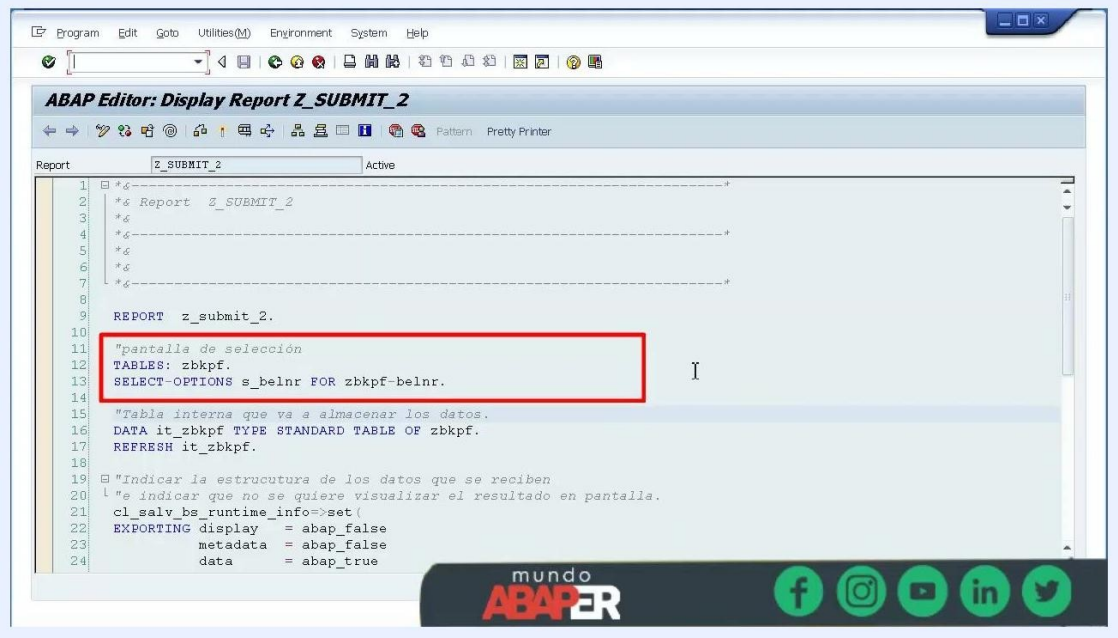
Step 7: Define the selection screen

Add the selection parameters your use case needs. For example, add a range for document numbers:

abap

```
TABLES: zbkpf.
SELECT-OPTIONS s_belnr FOR zbkpf-belnr.
```

You can add any other selection parameters your main screen requires.



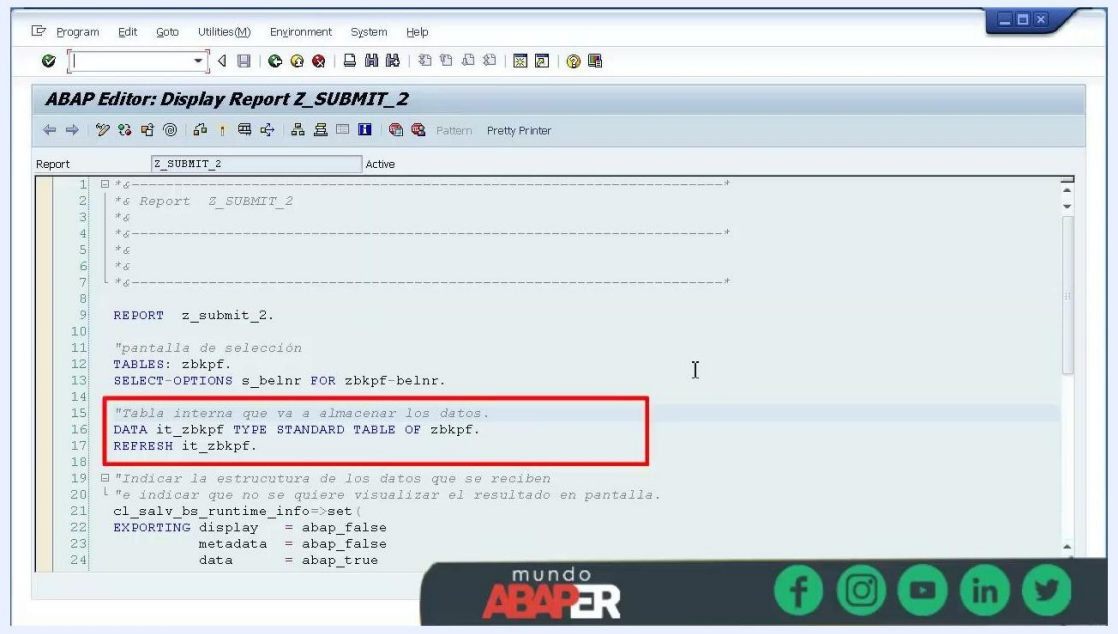
ABAP Editor showing the selection screen definition with TABLES: zbkpf and SELECT-OPTIONS s_belnr FOR zbkpf-belnr highlighted

Step 8: Declare the internal table for the results

Declare an internal table to store the output of the submitted ALV report. Base it on the structure or table you identified earlier, for example ZBKPF.

abap

```
DATA it_zbkpf TYPE STANDARD TABLE OF zbkpf.
REFRESH it_zbkpf.
```



ABAP Editor showing the internal table declaration DATA it_zbkpf TYPE STANDARD TABLE OF zbkpf highlighted

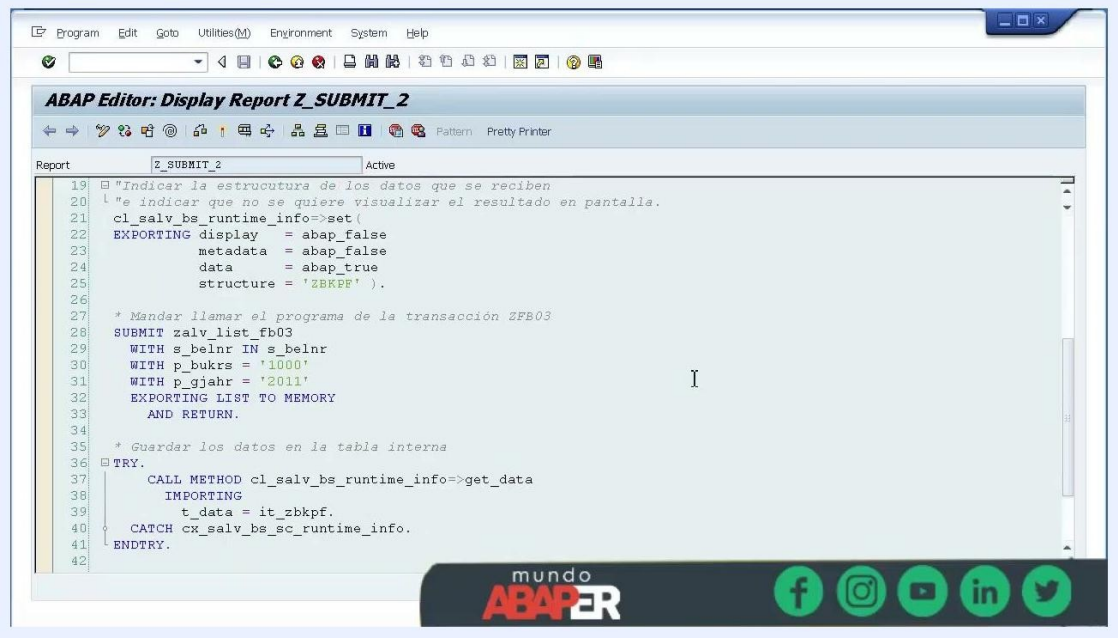
Step 9: Set up the runtime information to get data from the submit program

Use the class cl_salv_bs_runtime_info to configure how the submit runs. Set it so that the result and its selection screen are not displayed and only the data is retrieved. Also specify the structure (ZBKPF) that stores the data.

abap

```
cl_salv_bs_runtime_info=>set(
  EXPORTING
    display   = abap_false
    metadata  = abap_false
    data      = abap_true
    structure = 'ZBKPF'
).
```

This captures the ALV output directly into your internal table structure.



ABAP Editor showing the cl_salv_bs_runtime_info⇒set method with EXPORTING parameters display = abap_false, metadata = abap_false, data = abap_true, structure = 'ZBKPF'

Step 10: Submit the ALV report and export the output to memory

Use the SUBMIT statement to call the ALV report program, for example zalv_list_fb03. Pass the required selection parameters:

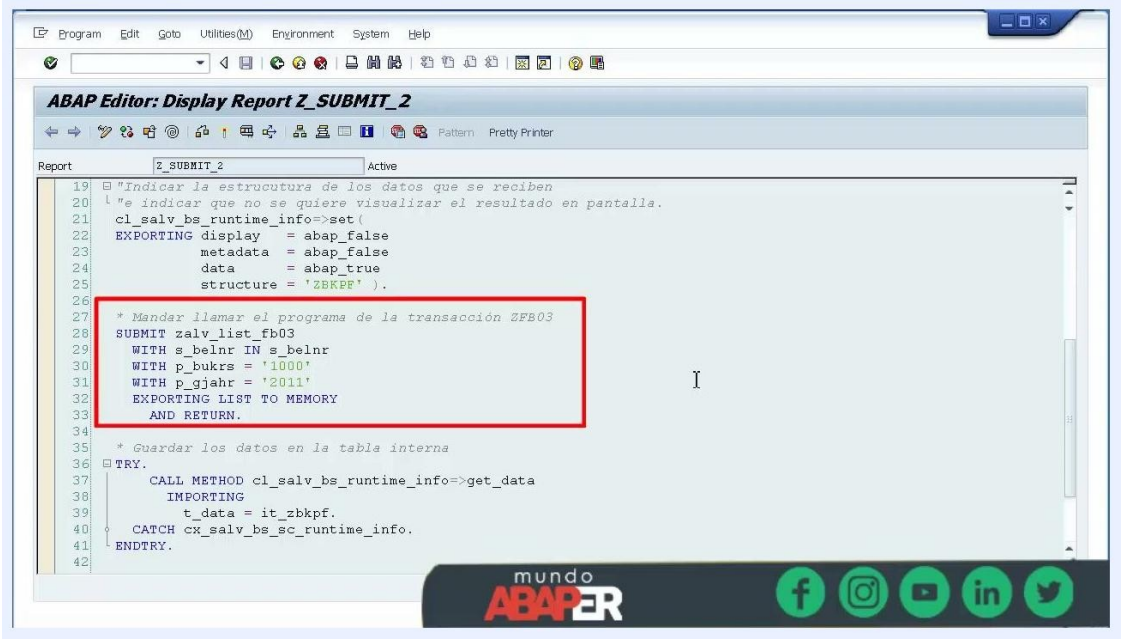
- s_belnr for the document number range
- p_bukrs for the company code, for example '1000'
- p_gjahr for the fiscal year, for example '2011'

Add EXPORTING LIST TO MEMORY AND RETURN so the output is stored in memory and not displayed.

abap

```
SUBMIT zalv_list_fb03
  WITH s_belnr IN s_belnr
  WITH p_bukrs = '1000'
  WITH p_gjahr = '2011'
  EXPORTING LIST TO MEMORY
  AND RETURN.
```

Generated with docsie.io



ABAP Editor with the SUBMIT statement highlighted, showing the parameters and EXPORTING LIST TO MEMORY AND RETURN

Step 11: Retrieve the data from memory into the internal table

After the submit, call the method cl_salv_bs_runtime_info=>get_data to transfer the data from memory into your internal table, it_zbkpf. Wrap the call in a TRY...ENDTRY block to handle exceptions.

abap

```
TRY.
  CALL METHOD cl_salv_bs_runtime_info=>get_data
    IMPORTING t_data = it_zbkpf.
CATCH cx_salv_bs_sc_runtime_info.
ENDTRY.
```

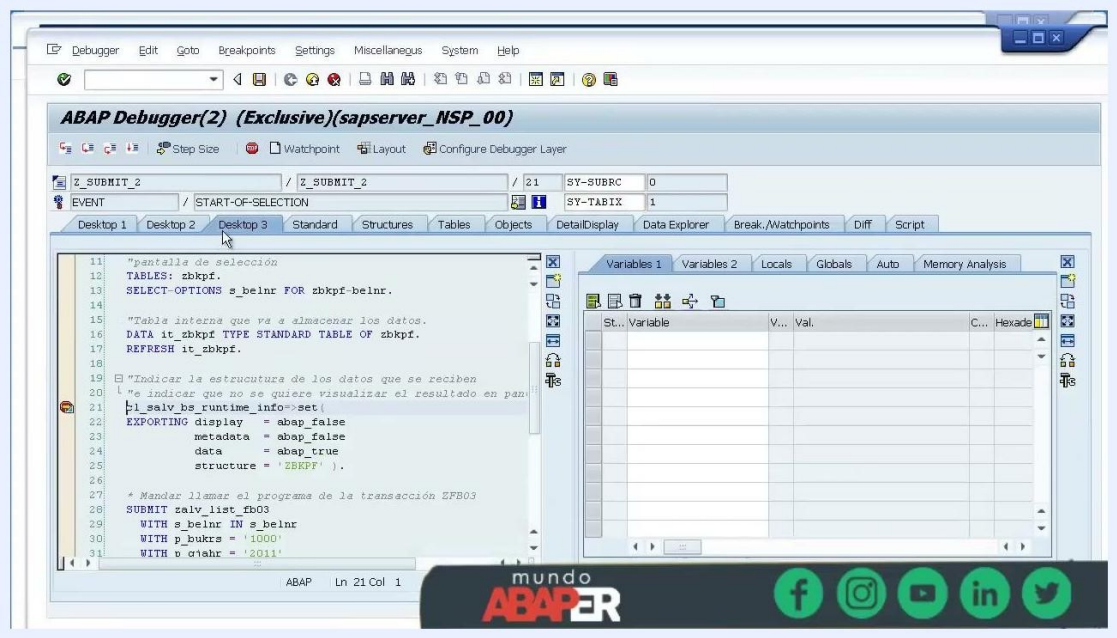
The data from the ALV report is now available in your program for further processing.

Step 12: Clear the runtime information

Finish by clearing the runtime information. This prevents memory leaks and leftover data, and it is a best practice.

abap

```
cl_salv_bs_runtime_info=>clear_all( ).
```



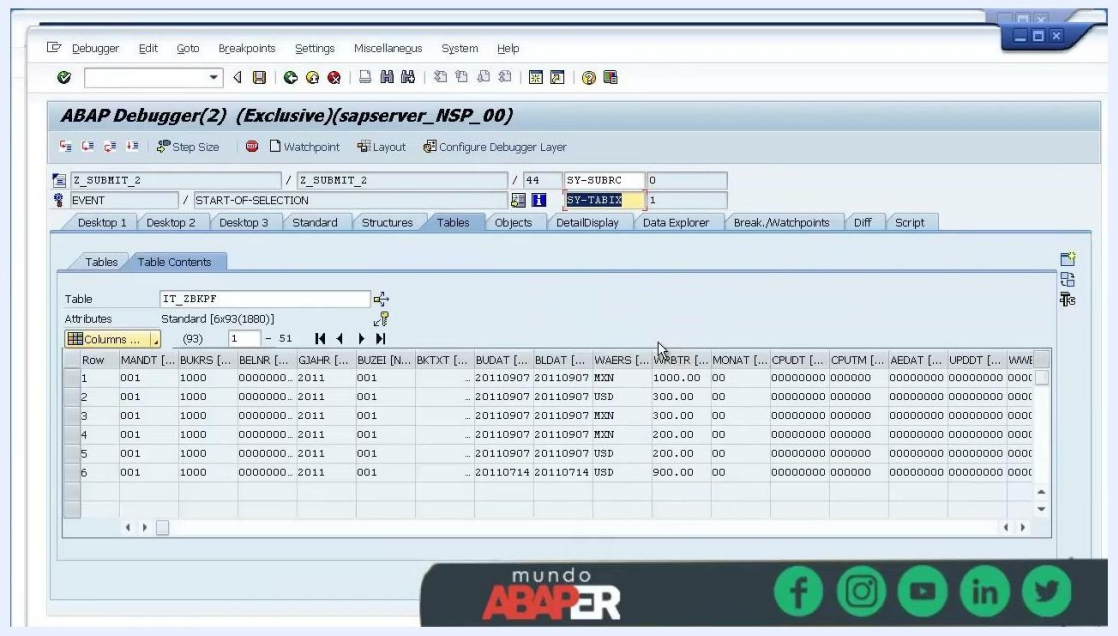
ABAP Editor showing the complete sequence: set runtime info, submit, get_data, and clear_all methods

Step 13: Debug the data retrieval

Use the ABAP Debugger to step through the code and verify each stage:

- Confirm the runtime info is set correctly.
- Make sure the submit runs with the correct parameters.
- Check that the internal table `it_zbkpf` is empty at first.

- After get_data runs, verify that it_zbkpf contains the expected data.



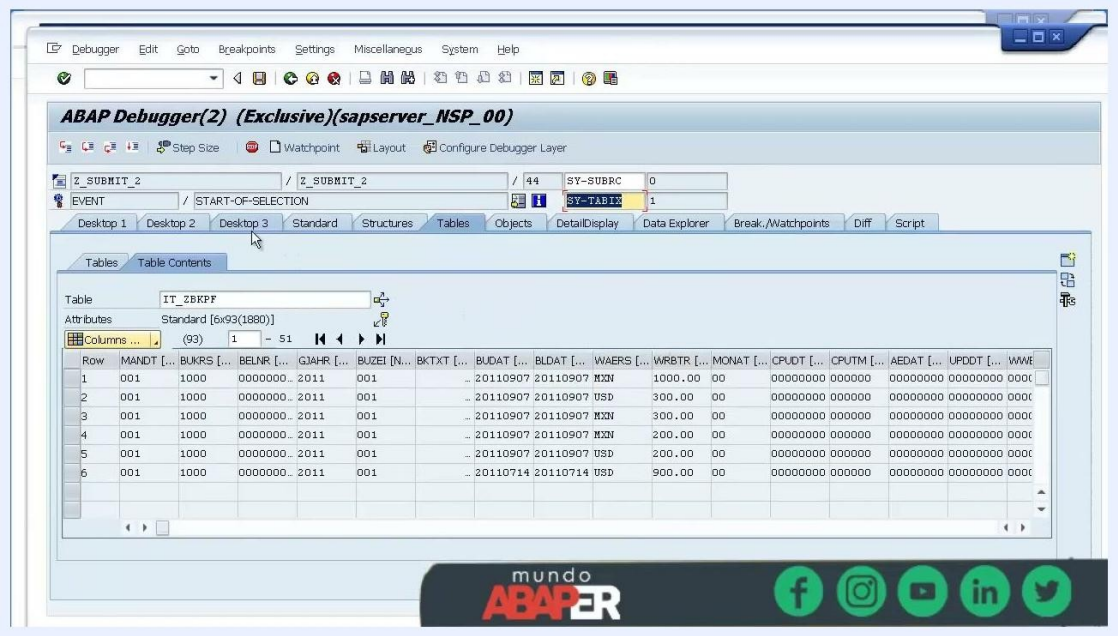
ABAP Debugger stopped at the point where the runtime info is set, with variable watch windows visible

Step 14: Verify the retrieved data in the ABAP Debugger

1. Open the ABAP Debugger after you run your program.
2. Go to the **Tables** tab and select **Table Contents**.
3. Locate the internal table IT_ZBKPF.
4. Confirm that it contains the expected fields and values.

Expected fields include MANDT, BUKRS, BELNR, GJAHR, BUZEI, BUDAT, BLDAT, WAERS, and WRBTR. Example values are MANDT = 001, BUKRS = 1000, BELNR = 0000000001, GJAHR = 2011, WAERS = MXN, and WRBTR = 1000.00.

Use the navigation controls to scroll through all rows and columns and check that the data is complete and accurate.



ABAP Debugger showing the IT_ZBKPF table with multiple rows and columns, including document numbers, company code, fiscal year, amounts, and currencies

Summary

You confirmed that the program behind your transaction is a report and identified the structure or transparent table used in the ALV output. If needed, you created a Z structure with matching field names and data elements. You then set the runtime information, submitted the report with EXPORTING LIST TO MEMORY AND RETURN, read the data into it_zbkpf with get_data, cleared the runtime information, and verified the result in the ABAP Debugger.

What's next

- Process the data in it_zbkpf in your program.
- Move your code into includes and a proper program structure instead of keeping everything in one program.
- Repeat the structure check from Step 4 for each ALV report you want to process with SUBMIT.