

SAP Summer School

Introduction to ABAP Workbench

internal

1. Packages and Reports
2. Development Objects. Active and Inactive
3. How to debug ABAP Programs
4. Optional exercise

1 Packages and reports

➔ navigate to your \$TMP package (local objects in se80)


1. Create a new package using the naming convention **TEST_INTERNSHIP2023_<your initials>** (by right-clicking on the super-package and selecting “New -> Package”). In this package you will store all your development artifacts during the course.

2. In a new session (transaction /ose80) ,open the ABAP Program **Z01_ABAP2023_SAMPLE** and copy it to you own package, under the name **Z01_ABAP2023_< your initials >**

3. Activate the program.

The program can be activated using Ctrl+F3 or 

4. Execute the program:

a. By clicking on the wrench () symbol, or




b. By pressing F8

What does the program do? Experiment with the program and have a look at the code to get an understanding of how it operates.

2 Development objects. Active and inactive versions

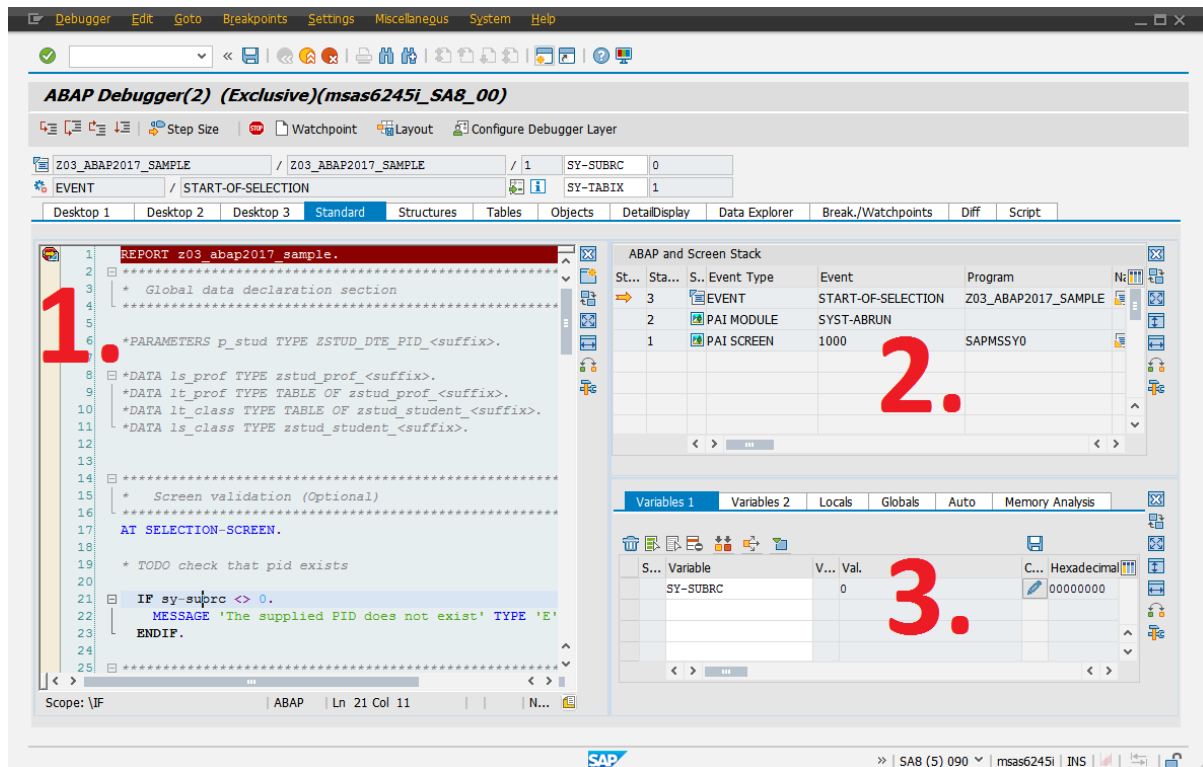
While it is being implemented, any ABAP development artifact is in an **inactive** state. However, in order to be executable in the ABAP runtime environment, the object has to be **activated** first. During activation syntax and other consistency checks are performed, after which a runtime object is generated. If the object is edited once again, an inactive copy is created. This copy can be edited without affecting the active runtime object (until a new activation takes place).

Display & Edit Mode. Activating Objects.

1. Open the source code of program **Z01_ABAP2023_< your initials >** in the Object Navigator
2. Switch from *Display* to *Edit Mode* by clicking the pen () icon, or by pressing *Ctrl+F1*
3. On line 8, change the name of variable *p_ord2* to *p_ord*. Save your changes (*Ctrl+S*) and perform a syntax check with *Ctrl+F2* or using the icon 
4. Correct any remaining syntax error. Afterwards the program can be activated using *Ctrl+F3* or 

3 How to debug ABAP programs

The ABAP debugger is a straightforward tool that helps you analyze what is wrong with your programs. The layout is divided in different tabs: (1) standard desktop with your source code, (2) the call stack and (3) the variables and their values. By double clicking on the variables in (3), you can analyze their content. We navigate back to the standard desktop using the Back button. (🏠)



In order to enter this perspective, you simply have to add a breakpoint to your active source code and run your application. You have four options for tracing your code:



- F5 – single step mode
- F6 – execute
- F7 – return
- F8 – continue.

4 Optional Exercise

Create a report where a user has the possibility to select flights based on his needs.

Based on:

- Price (lower than X)
- Airline Company
- Desired class (economy, business or first class). It is assumed that there are free places on the selected class.

A checkbox is defined as selection criteria for the user and in case the checkbox is marked, just the first available flight is displayed (not in a list).

Extra:

- City from
- City to
- Flight date
- The airline Id must be validated at input, meaning that it should already be existing in the database

The user has the possibility to see the list, sorted by price (ascending), details for the available flights: flight date, flight number, company, price, airport from, departure time, airport to, arrival time, flight duration.

In case there are no flights to show, a warning message will be displayed that there are no flights.

Extra:

- Display the remaining seats for each class.
- In case the checkbox is selected and the plane is overbooked (more passengers than seats), display an error message.

Technical requirements:

- Flight tables: SCARR, SPFLI, SFLIGHT
- Please use the name: Z<initials>_EX1 for the report.
- All the objects will be created under your local package.
- For the fetching of data from the database tables, use a new function module (zfm<initials>_sflight_<user>).
- For the validation of the airline id, create a subroutine.
- Feel free to create the texts for messages and parameters.
- You can use the following syntax for list display(if you want):
cl_demo_output=>display(
 EXPORTING
 data = lt_data "table with data
 name = 'Display' "description
).