How to debug a background job (BJOB)

In the work of a SAP Consultant comes time when you need to debug a program in the background mode or a running BJob.

There is a simple trick how to do this:

1. Set a breakpoint in ABAP report / FM that you want to debug.

2. Create a background JOB in SM36 transaction or use any other available methods. Set a future time for start (enough +10 min).

Program	Edit	Goto	System	Help

Execute   F8
Execute and Print   Ctrl+P
Execute in Background   F9
Exit   Shift+F3

R   k transfer documents: by material

Data

Material		to		
Plant		to		

Options

☑ Process sales documents
☐ Process stock transfer docs
   |__ At item level   ●
   |__ At schedule line level   ○

☑ Unconfirmed documents required
☐ Simulation

Sort order

	Priority	
Document category	1	
|__ Prioritize sales documents		●
|__ Prioritize stock transfer documents		○
Delivery priority	2	
Date	3	
|__ Sort item by date of creation		●
|__ Sort by delivery date of earliest schedule line		○
Document number	4	
Document item	5	

Rescheduling sales and stock transfer documents: by material

OE1(2)/100 Background Print Parameters

Output Device	
Number of Copies	1

Page Area

- ● Everything
- ○ Page 0 to 0

Properties

Print Time	SAP spool only for now

Properties

Sort order

	Priority
Document category	1
|__ Prioritize sales documents	●
|__ Prioritize stock transfer documents	○
Delivery priority	2
Date	3
|__ Sort item by date of creation	●
|__ Sort by delivery date of earliest schedule line	○
Document number	4
Document item	5

Rescheduling sales and stock transfer documents: by material

⊕ 🗐 ⓘ

📤 OE1(2)/100 Start Time   ✕

| Immediate | Date/Time | After Job | After Event | Operation Mode | ➡ |

Date/Time

After Job

Operation Mode

After Event

✅ Check   💾   ❌

Document number   4
Document item   5

3. In SM37 transaction, put the cursor on the job line, type '**JDBG**' in the command line (no '/') and press enter.

4. The JOB will start in debug mode.

(/hs)ABAP Debugger(2) (Exclusive)(ch00xxxxx_CE1_00)

Step Size	Watchpoint	Layout	Configure Debugger Layer	

| SAPLSPOO | / | LSPOOU07 | / 14 | SY-SUBRC | 0 |
| FUNCTION | / | RSPO_TEST_LAYOUT | | SY-TABIX | 2 |

Desktop 1 | Desktop 2 | Desktop 3 | Standard | Structures | Tables | Objects | DetailDisplay | Data Explorer | Break./Watchpoints | Diff | Script

```
 8   *"      EXPORTING
 9   *"        VALUE(ANSWER)
10   *"        VALUE(LAYOUT_EXISTS)
11   *"        VALUE(LAYOUT_TYPE) TYPE  RSPOFMTTYF
12   *"----------------------------------------------
13
14   IF devicetype IS INITIAL.
15      SELECT SINGLE * FROM tsp03 WHERE padest = device.
16      IF sy-subrc <> 0.
17        answer = ' '.                  "Gerät existiert nicht
18        EXIT.
19      ENDIF.
20
21      devicetype = tsp03-patype.
22   ENDIF.
23
24   PERFORM test_layout USING devicetype layout
25                       CHANGING layout_type.
26
27   IF sy-subrc = 0.
28      answer = 'X'.
29      layout_exists = 'X'.
30   ELSE.
31      layout_exists = ' '.
32      answer = ' '.
33      CLEAR layout_type.
34   ENDIF.
35
36   ENDFUNCTION.
37
38
```

ABAP and Screen Stack

Sta...	Stac...	S..	Event Type	Event	Program	Na...	Include
➡	9		FUNCTION	RSPO_TEST_LAYOUT	SAPLSPOO		LSPOOU07
	8		FORM	FIT_PARAMS	SAPLSPRI		LSPRIF14
	7		FORM	GET_PRINT_INFO	SAPLSPRI		LSPRIF11
	6		FUNCTION	GET_PRINT_PARAMETERS	SAPLSPRI		LSPRIU01
	5		FUNCTION	PRINT_REPORT	SAPLSPRI		LSPRIU05
	4		FORM	%_PRINT_REPORT_%	RSDBRUNT		RSDBRUNT
	3		EVENT	SYSTEM-EXIT	SDV03V02		<SYSINI>
	2		PBO MODULE	SYST-ASUBM			
	1		PBO SCREEN	1000	SAPMSSY0		

Variables 1 | Variables 2 | Locals | Globals | Auto | Memory Analysis

St...	Variable	V...	Val.	C...	Hexadecimal Value

S | ABAP | Ln 14 Col 1 | NUM