

Debugging ABAP Program

By

René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)

Topic Covered :

1. What is ABAP Debugging?
2. Important transaction code in ABAP Debugging
3. Breakpoints
4. Debugging Screen
5. Watchpoints
6. Changing Variable Values
7. Real time Debugging :
 - a) Debugging pop up screen
 - b) Debugging of Enhancement
 - c) Debugging an error message

1. What is ABAP Debugging?

Debugging is when you stop a program or transaction at a certain point, so you can walk step by step through the ABAP programming to see where an error occurs.

But as functional consultant you need basic ABAP skills to be able to do this, Most good functional consultants will have basic debugging skills and this is very very important because it reduces the excessive independence to ABAP teams.

Most of the time, part of our workload is to resolve errors that occur during the execution of certain transactions. However, if these errors message cannot be resolved by setting up the master data or transactional data (i.e.: error message without any meaning) you need to access the program in debug mode (by typing “/h” in the transaction bar and then executing the transaction), which will allow us to analyze the program and see where the problem lies, and thus save time.

Debugging ABAP Program

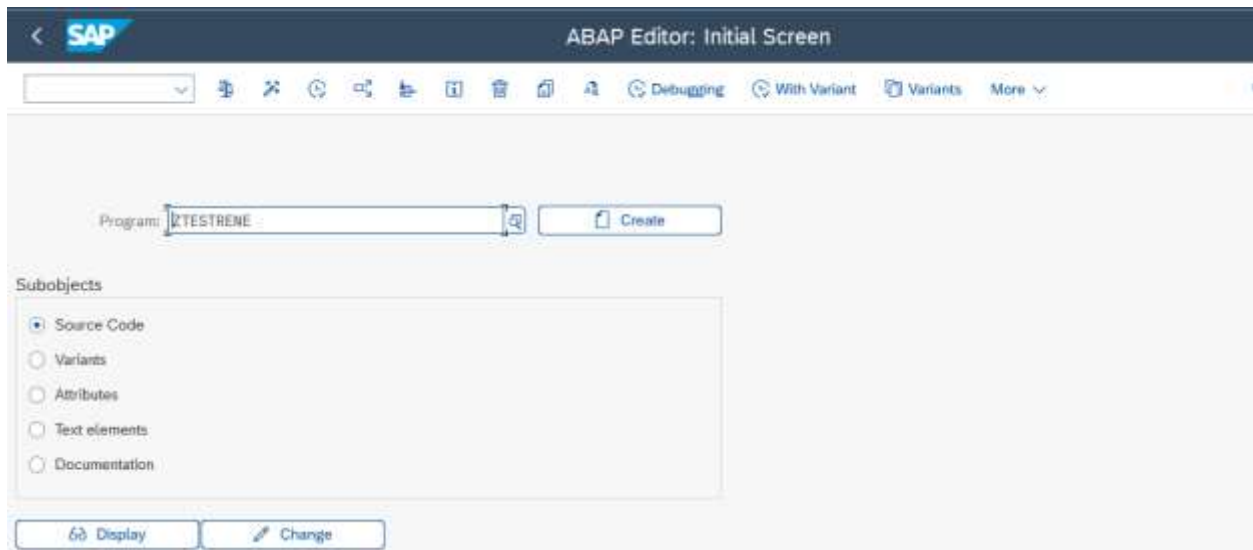
By

René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)

2. Important transaction code in ABAP Debugging :

Here we have two important transaction code: SE38/SE37

SE38: is a transaction code that allows you to display/change/create ABAP Program.



SE37: is a transaction code that allows you to display/change/create Function Module.

Function Modules: are set of codes that can be reused by others programs and FM can be executed independently.

Let's take an example: Function Module: (**SD_ROUTE_DETERMINATION**)

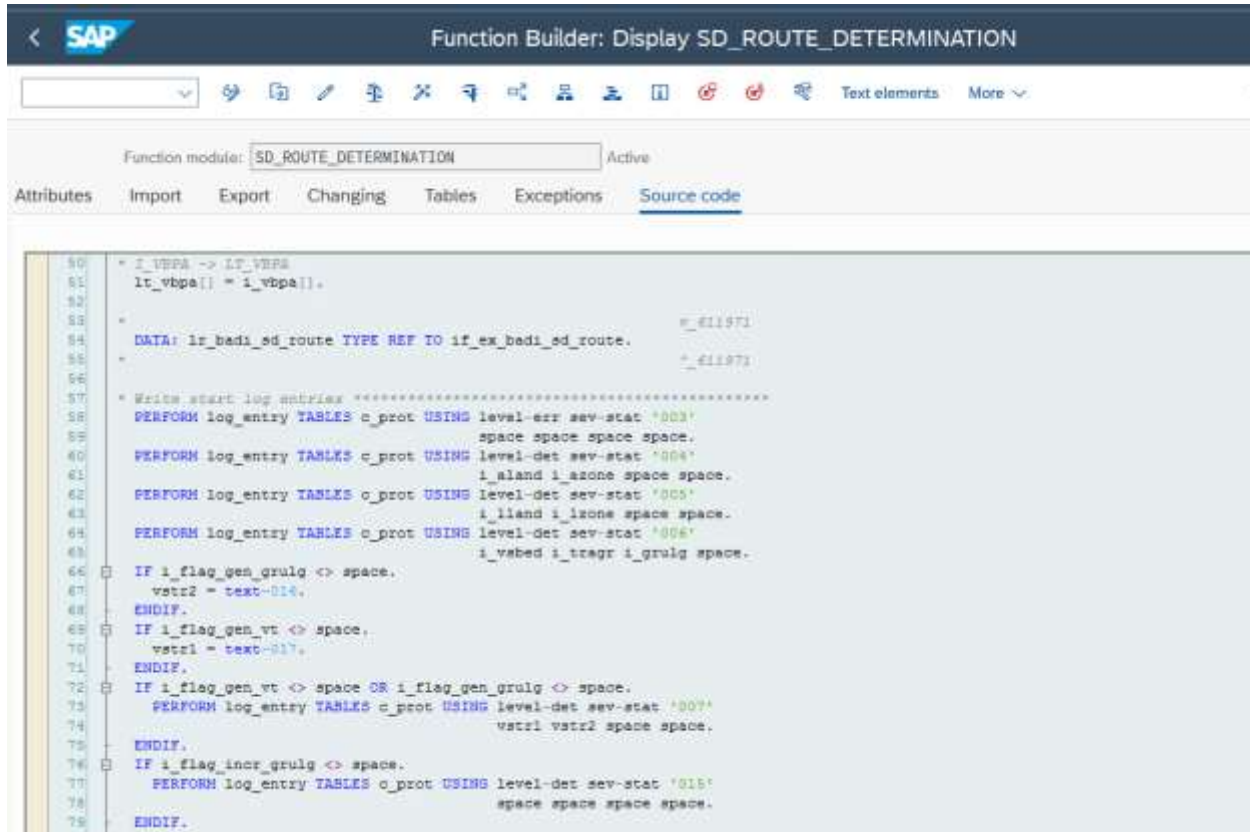
This FM is used for the logic of route determination

Now let's open this FM with help of the SE37 transaction code

Debugging ABAP Program

By

René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)



```
50 * I_VBPA -> IT_VBPA
51 lt_vbpa[] = i_vbpa[].
52
53
54 DATA: lt_badi_sd_route TYPE REF TO if_ex_badi_sd_route.
55
56 *
57 * Write start log messages *****
58 PERFORM log_entry TABLES c_prot USING level-err sev-stat '003'
59 space space space space.
60 PERFORM log_entry TABLES c_prot USING level-det sev-stat '004'
61 i_aland i_azone space space.
62 PERFORM log_entry TABLES c_prot USING level-det sev-stat '005'
63 i_liand i_lzone space space.
64 PERFORM log_entry TABLES c_prot USING level-det sev-stat '006'
65 i_vbed i_tragr i_grulg space.
66
67 IF i_flag_gen_grulg <> space.
68   vstr2 = text-014.
69 ENDIF.
70 IF i_flag_gen_vt <> space.
71   vstr1 = text-017.
72 ENDIF.
73 IF i_flag_gen_vt <> space OR i_flag_gen_grulg <> space.
74   PERFORM log_entry TABLES c_prot USING level-det sev-stat '007'
75   vstr1 vstr2 space space.
76 ENDIF.
77 IF i_flag_incr_grulg <> space.
78   PERFORM log_entry TABLES c_prot USING level-det sev-stat '015'
79   space space space space.
80 ENDIF.
```

3. Breakpoints

In ABAP debugging, the notion of breakpoint is a very important notion because the breakpoint is used to interrupt the processing of the program and launch the debugger.

When you are in the ABAP editor (SE38) the two breakpoint icons are displayed at the top of the page



Left session breakpoint and right external breakpoint

To set a breakpoint in an ABAP program, simply select the line of executable code and click on one of the icons above.

Debugging ABAP Program

By

René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)

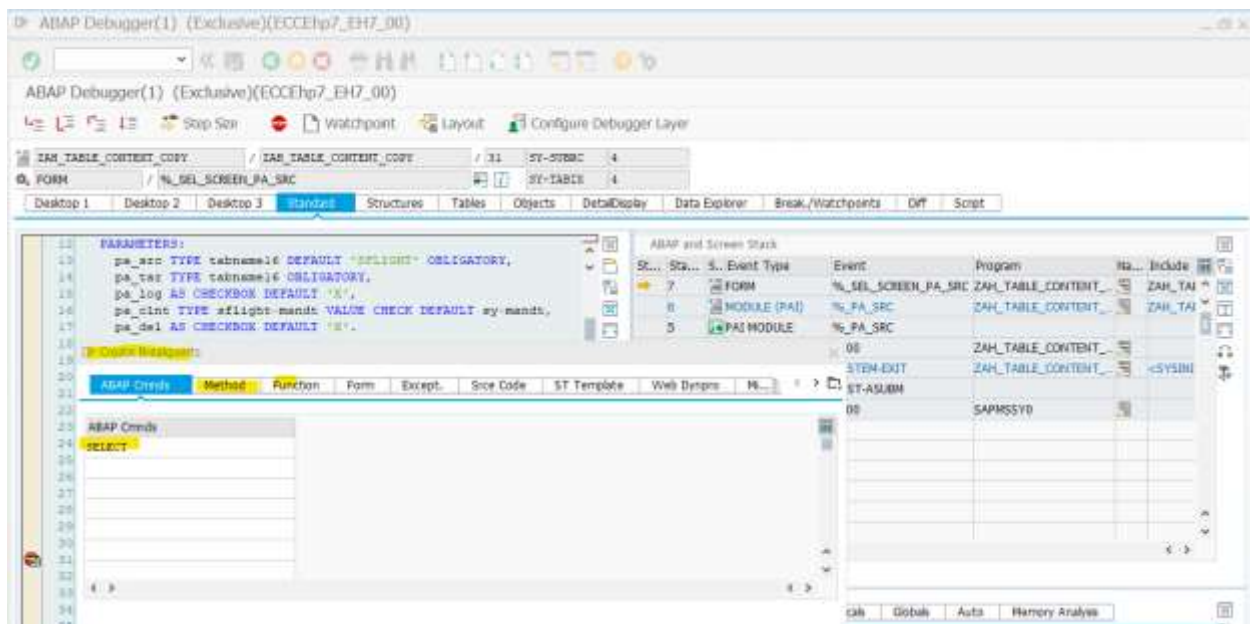
Now let's discuss the meaning <<Session breakpoint>> and <<external breakpoint>>

Session Breakpoint: Specific to a particular ABAP user session. If the user session is ended (log off sap system) all session breakpoints get deleted.

External Breakpoint: They are applied to the current user session as well as future user sessions.

The other thing to remember here is that you can put a breakpoint on almost anything, for example if you think your problem is with specific ABAP statements (IF.....ENDIF, SELECT, LOOP); function modules or methods .

For that you have to go to the breakpoint creation menu



Important: An ABAP program can contain hundreds or even thousands of lines of code, so it is impossible to go through the program line by line to find the problem. It is therefore essential to master the use of breakpoints because if you think the problem is at a certain level in your program, you can simply place your

Debugging ABAP Program

By

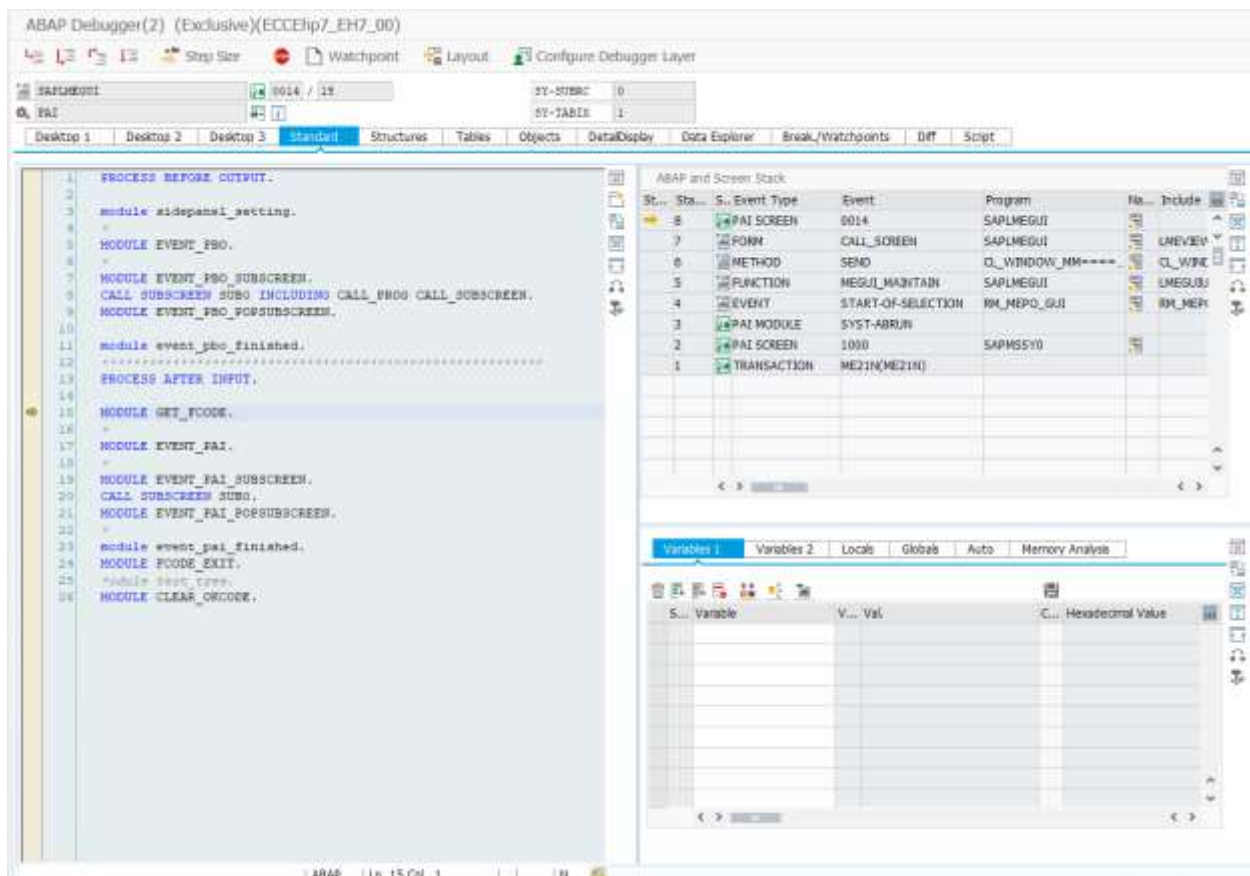
René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)

breakpoint at that level and SAP will stop processing the program at that level and run the debugger.

4. Debugging Screen

To explain the debugging screen, I will use the transaction (ME21N), typing “/h” in the transaction bar, then running the transaction. This will allow us to access the program in debug mode.

You will probably see the screen below



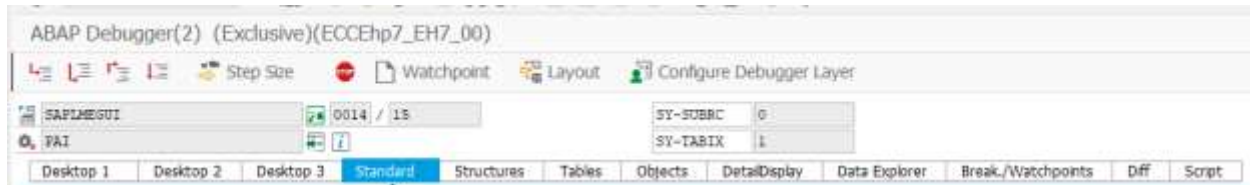
Let us discuss this section by section

Debugging ABAP Program

By

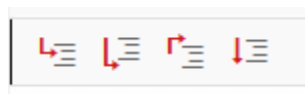
René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)

At the top of the debugging screen



Here you can easily read the name of the program and also see the event being processed (function/method).

To take the process forward step by step you can use the icon strip:



Now let us discuss one by one:



(The F5 key): Allows you to execute the code line by line



(The F6 key): This icon allows you to execute the code line by line without going into the code portions called via MF / routine / other programs.



(The F7 key): if you are inside of MF or method or perform it will take you out of that module and return to the main program.



(The F8 key): Allows the code to be executed in one go until the next breakpoint set if any otherwise the debugging is switched off and the program is executed completely.

SY-SUBRC	0
SY-TABIX	1

SYST structure:

Structure fed and updated throughout the processing.

Debugging ABAP Program

By

René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)

We will see here 2 important SYST structures:

SY-SUBRC: is SAP ABAP system field and contains a return code of ABAP statements. This value is used to determine the status of the execution of an ABAP statement.

Important: if SY-SUBRC is 0 the ABAP statement has been executed successfully if the value is different from 0 then the statement has raised an error or warning.

SY-TABIX: returns the number of the current line in a processing loop

5. Watch-Points

Can be set on a variable you want to monitor to stop program execution when the contents of that variable change, but only you need to ensure that the variable is declared globally in a program

What's even more interesting about watchpoints is that you can set them to fire when specific conditions are met (for example, you give your variable a fixed value) instead of firing every time the variable changes, because your variable may change at several places in the program.

Most of the time, an internal table contains thousands of rows and if we need to debug a particular row, for example the item number which is in the 300th row of an internal table, knowing that this internal table contains more than 10,000 rows in this case, it is very difficult, if not impossible, to debug an internal table row by row to reach this particular row.

In order to save time, it is sufficient to define a watchpoint for the particular value of a field or variable.

How to use a watchpoints? Let's take 2 scenarios:

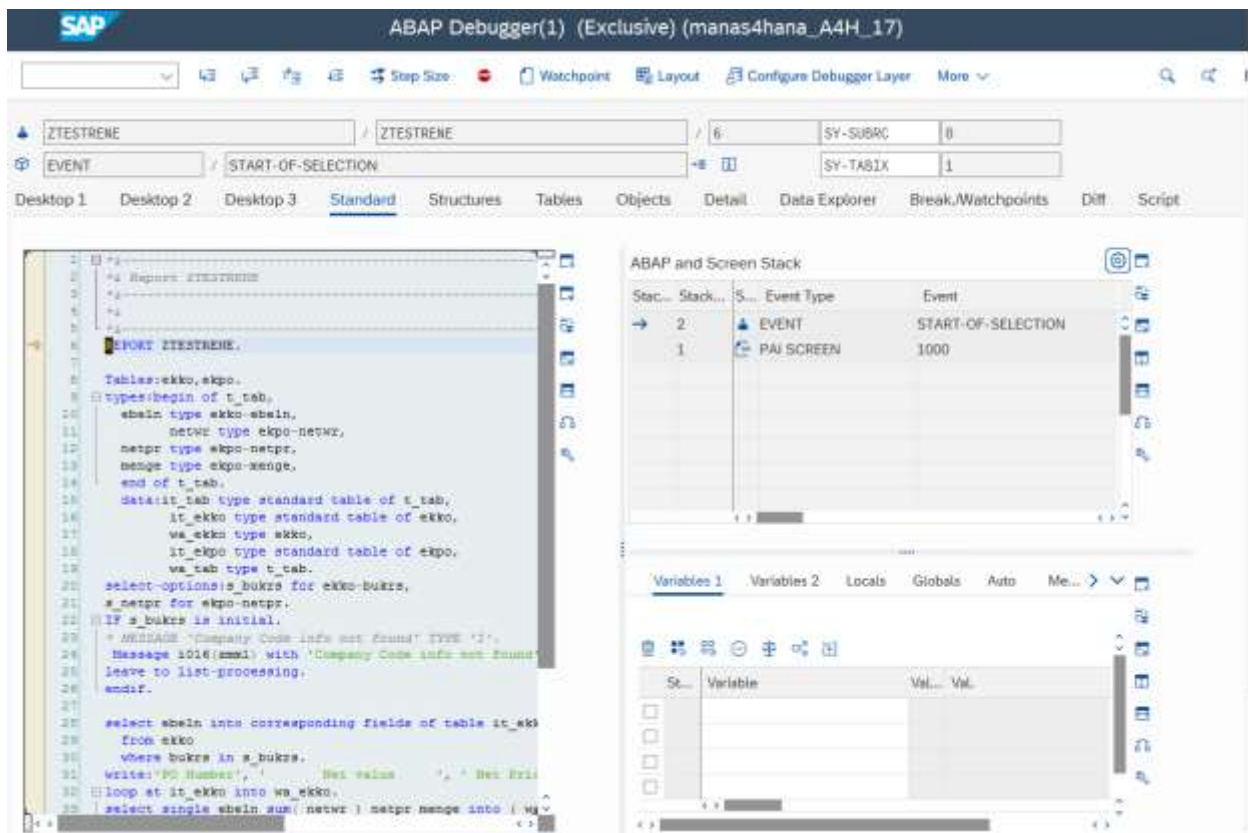
Debugging ABAP Program

By

René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)

Scenario 1: in our custom program we want to see the behavior of one particular variable we will ask SAP to stop the program when it reaches this variable for achieve our goal we will use watchpoints let's see how

Go to se38 and open your program in debug mode

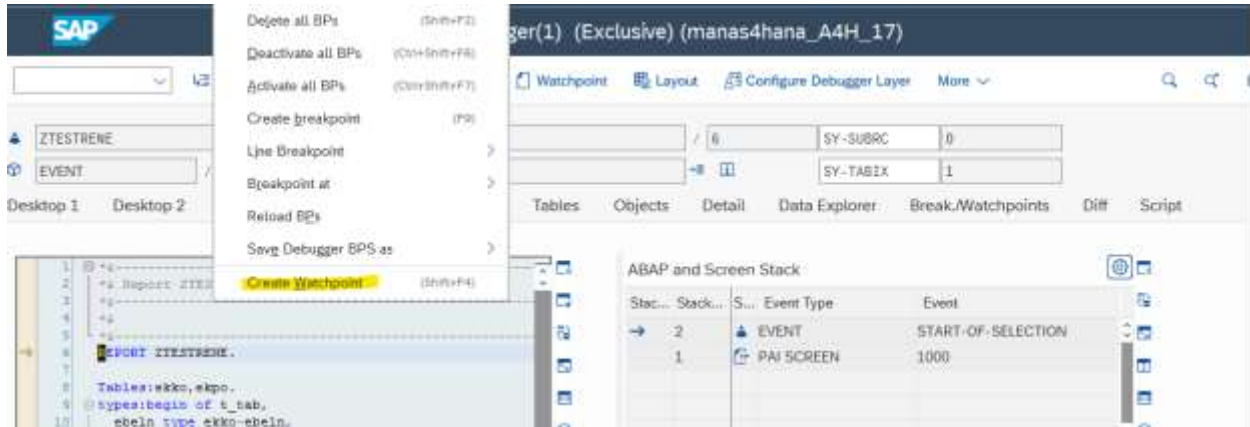


Create watchpoint

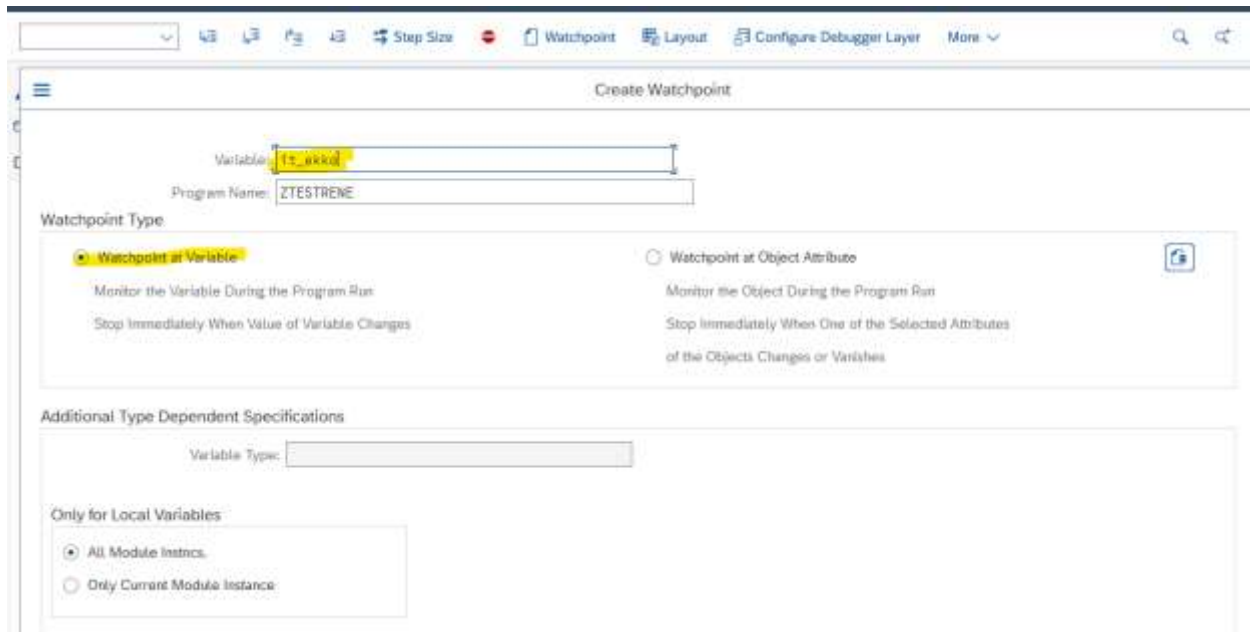
Debugging ABAP Program

By

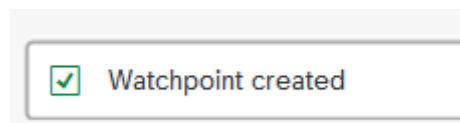
René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)



Put the name of variable you want to check and click on the green button



You will get this message

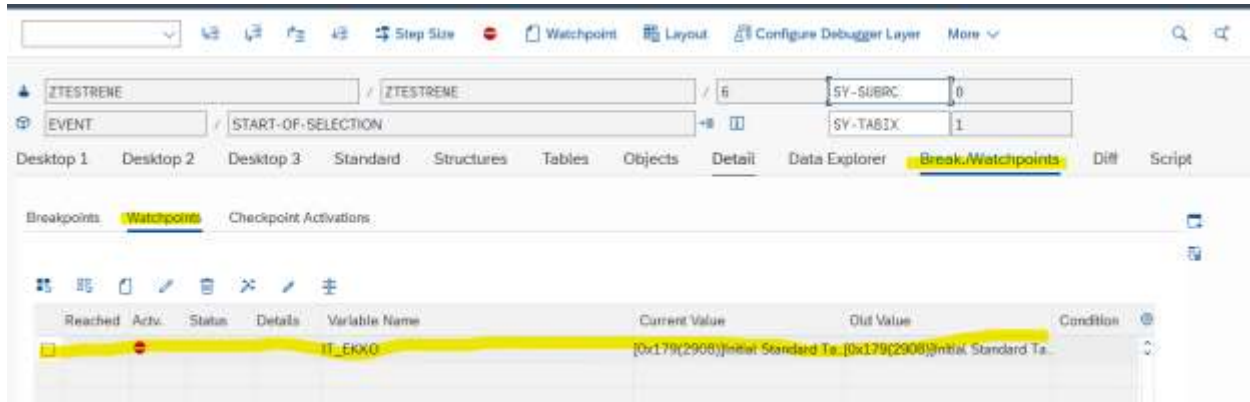


You can also check in watchpoints tab

Debugging ABAP Program

By

René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)



And execute the program (F8)

And finally you will get this confirmation message



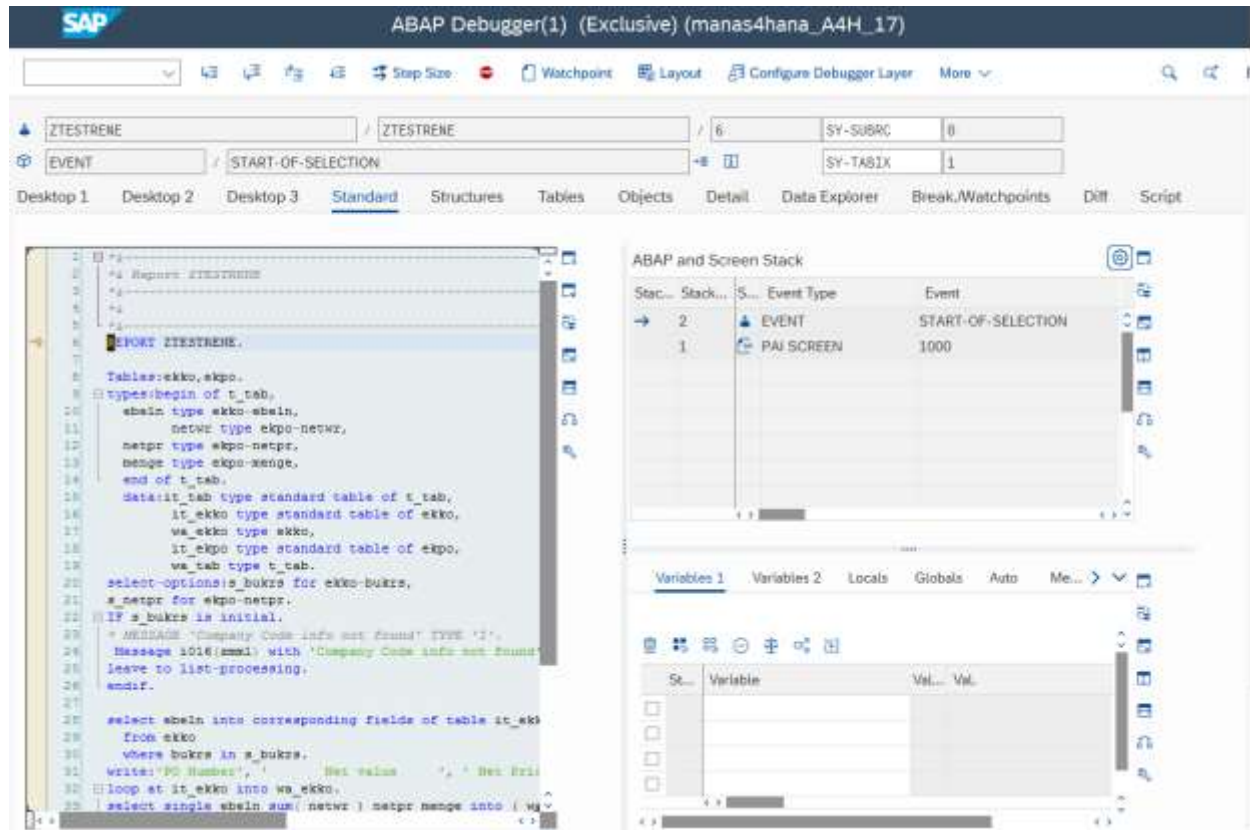
Scenario 2: For example, there are 386 entries in the internal table. Let's see how to analyse the 227th record, i.e. the PO number "4500001592".

Go to se38 and open your program in debug mode

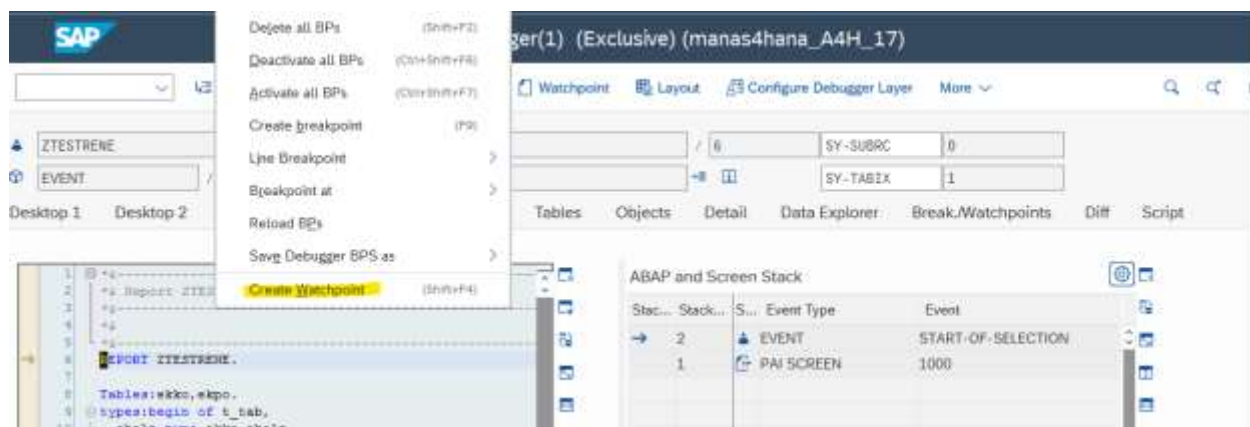
Debugging ABAP Program

By

René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)



Create watchpoint



Because we want to see a particular value, we will create a watchpoint but this time our variable will be a 'work area' with the field corresponding to our value.

Debugging ABAP Program

By

René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)

Create Watchpoint

Variable: WA_EKKO-EBELN

Program Name: ZTESTRENE

Watchpoint Type

☒ Watchpoint at Variable
Monitor the Variable During the Program Run
Stop Immediately When Value of Variable Changes

☐ Watchpoint at Object Attribute
Monitor the Object During the Program Run
Stop Immediately When One of the Selected Attributes of the Objects Changes or Vanishes

Additional Type Dependent Specifications

Variable Type:

Only for Local Variables

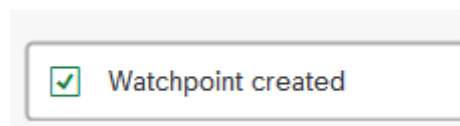
☒ All Module Instances
☐ Only Current Module Instance

No Additional Condition

Condition

Free Condition Entry: <= 4500001592

You will get this message



You can also check in watchpoints tab

ZTESTRENE / ZTESTRENE / 32 / SY-SUBRC 0

EVENT / START-OF-SELECTION / SY-TABIX 1

Desktop 1 Desktop 2 Desktop 3 Standard Structures Tables Objects Detail Data Explorer Break/Watchpoints Diff Sc

Breakpoints Watchpoints Checkpoint Activations

Variable Name	Current Value	Old Value	Condition	Type
WA_EKKO-EBELN			WA_EKKO-EBELN <= 4500001592	Global in a Program

Debugging ABAP Program

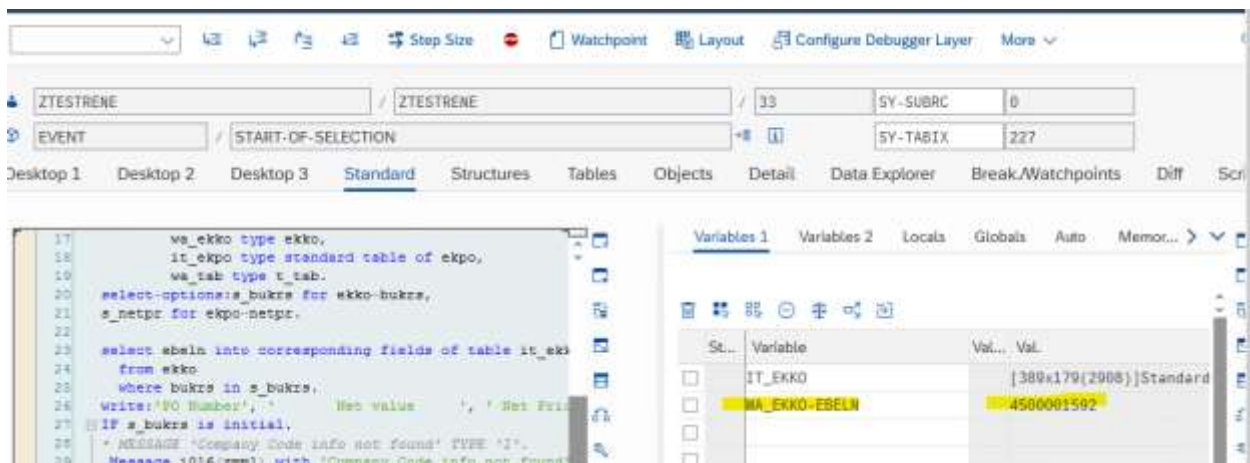
By

René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)

And execute the program (F8)

And finally you will get this confirmation message

✓ Watchpoint reached (WA_EKKO-EBELN)



6. Changing Variable Values:

One of the important features of the SAP debugger is the ability to change the values of variables "on the fly".

Sometimes, when we debug for testing purposes, we need to change the values of variables in order to test several scenarios or to see the behavior of the program.

And even during troubleshooting, changing the value of a particular variable can help solve the problem.

However, it is important to ensure that you have permission to do so, otherwise the system will not allow you to change a field. You should therefore contact the SAP security team to obtain the necessary permissions.

Debugging ABAP Program

By

René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)

Let's see how we can change the value

For the testing purpose we will change the PO number (from: 4500001592 to: 4500001630) to see the behavior of the program

Variables 1

St...	Val...	Val.	Ch...	Hexadecimal V
		[390x179(2908)]Standard Table		
		4500001592		340035003000

Click to pencil icon to make the field editable

ABAP Debugger(1) (Exclusive) (manas4hana_A4H_17)

Variables 1

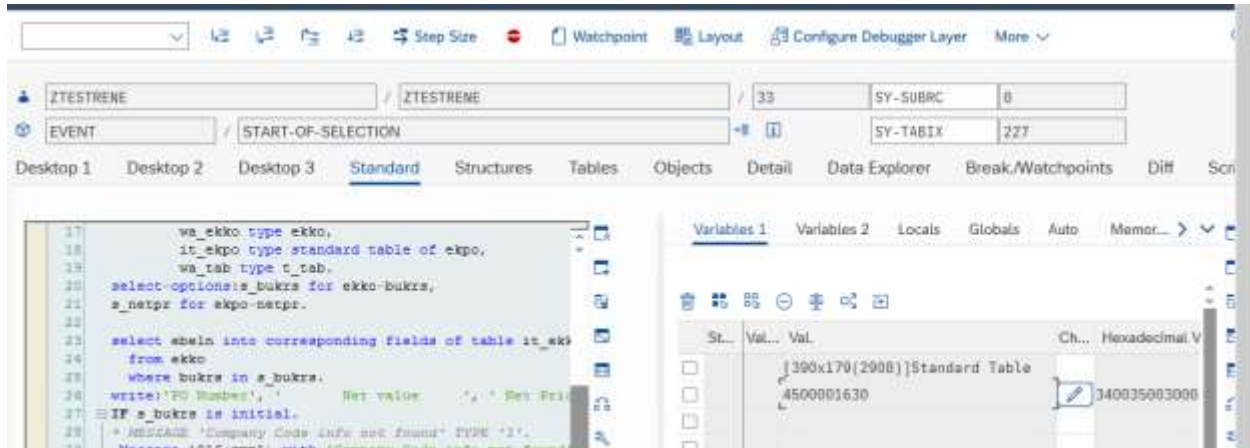
St...	Val...	Val.	Ch...	Hexadecimal V
		[390x179(2908)]Standard Table		
		4500001592		340035003000

Now we can change the PO number easily

Debugging ABAP Program

By

René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)



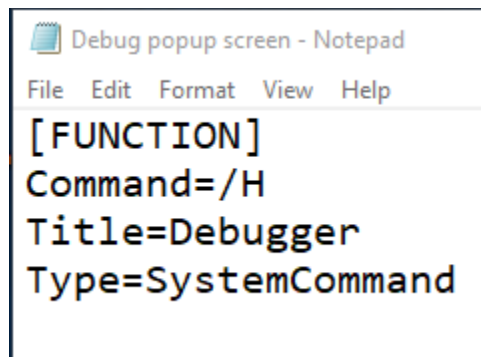
7. Real time Debbuging:

a) Debbuging pop up screen:

In SAP we encounter pop-up screens every day, but there are scenarios where a pop-up screen needs to be debugged to find the exact cause of the error.

One of the most important difficulties is that command line debugging is not possible if pop-up screens appear: **because you cannot edit /h**

What you need to do is save the line of codes below to a text file on your laptop or desktop.



Debugging ABAP Program

By

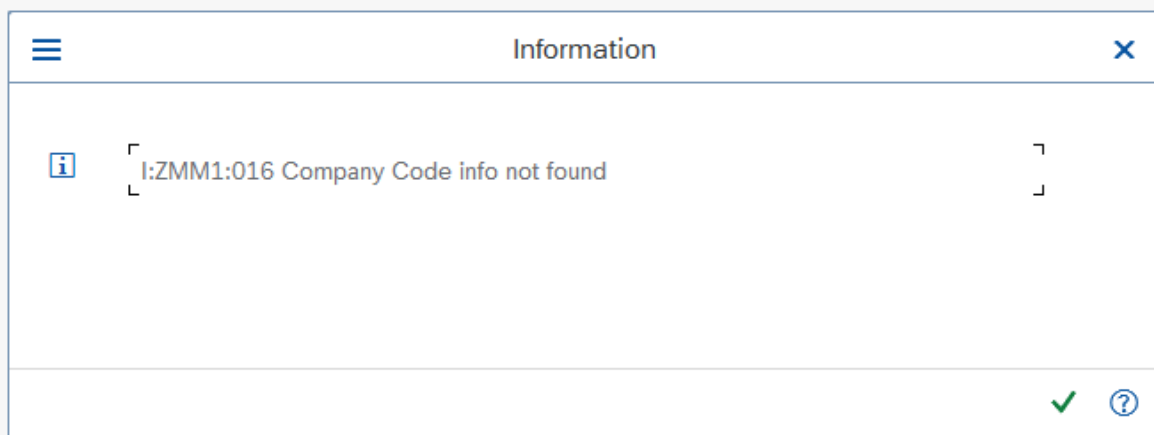
René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)

And when it's time to debug, just drag and drop the text file you saved on your desktop or laptop onto the pop-up screens.

Let's take an example

During a transaction, we get this pop up screen here below

Our task now is to find out exactly which line of the ABAP code contains this error by debugging.

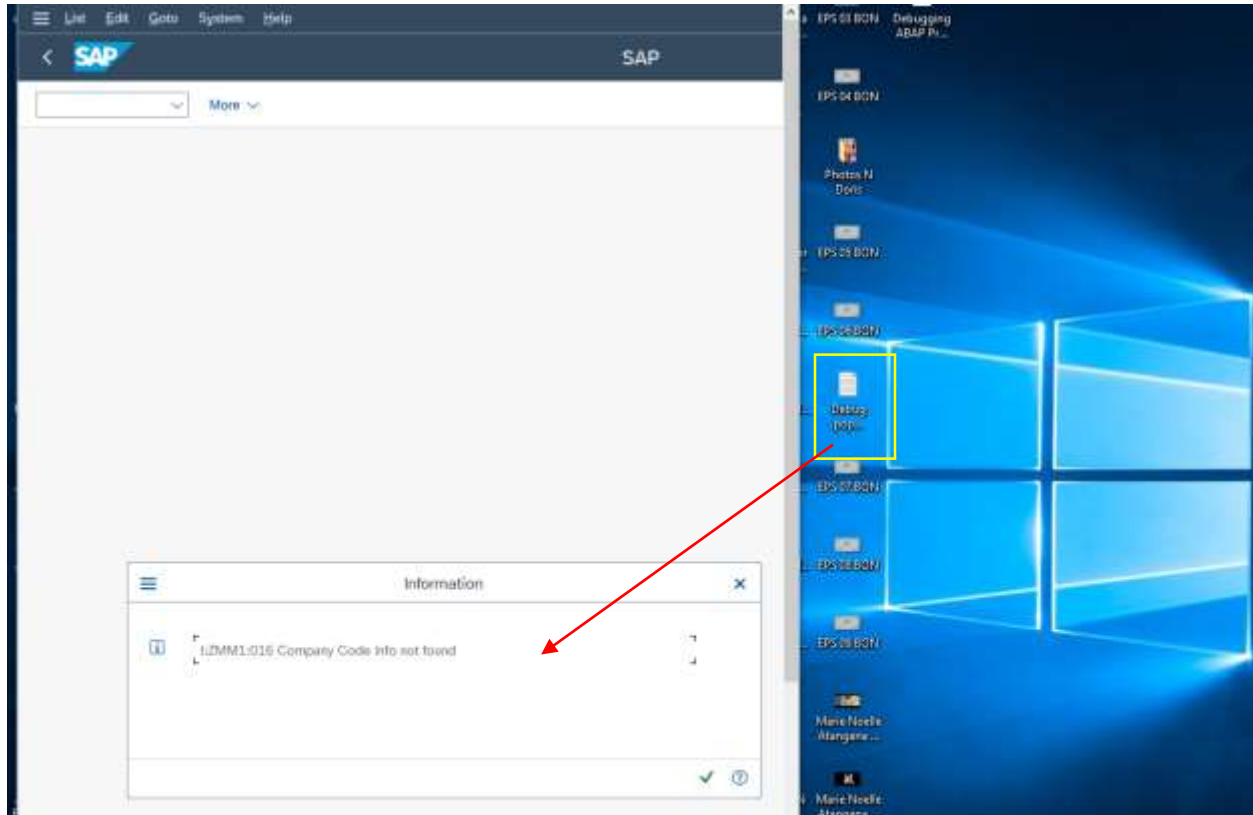


Just drag and drop the text file you saved on your desktop or laptop onto the pop-up screens.

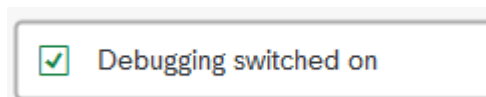
Debugging ABAP Program

By

René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)



You will get this message



And click on the green button

The program will stop exactly at the point where this error occurs.

Enhancements: enables customers/clients to add custom business functionality to sap standard software without changing at all the sap standard functionality (which is supposed to be unchanged).

Debugging ABAP Program


By

René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)

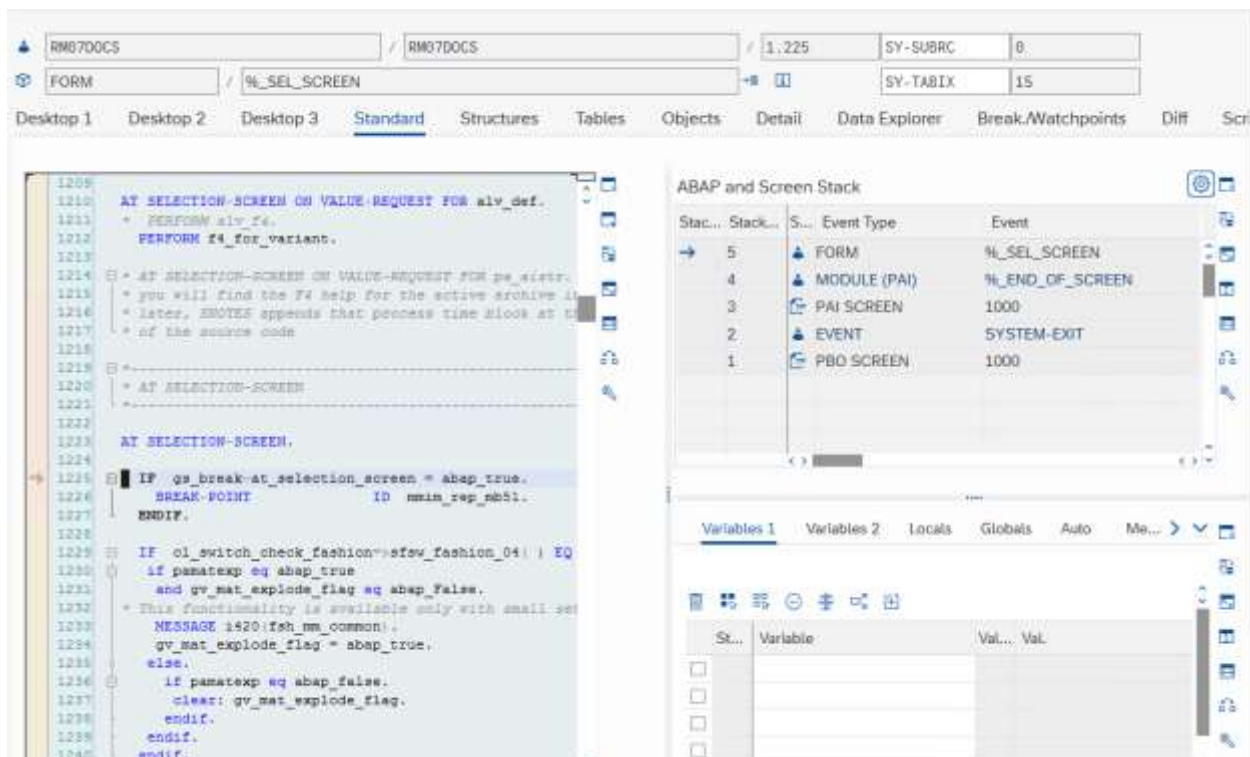
But to better manage the enhancements in the program and also to be sure not to change the standard functionality of the sap, the best thing to do is to copy a standard program and make the enhancements you want.

Now let's see how we can debugging enhancement

Two things you should keep in your mind before starting debugging enhancement:

- Icon snail : 
- And(the F5 key) button remember (The F5 key): Allows you to execute the code line by line

Go to se38 and open your program in debug mode



Debugging ABAP Program

By

René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)

Identify the snail icon in the program as shown below

Debugging ABAP Program

By

René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)

```
2161 IF cl_change IS BOUND.
2162     CALL METHOD cl_change->is_active
2163     RECEIVING
2164         rv_active = lf_active.
2165 ENDIF.
2166
2167 IF lf_active = 'X'.
2168     TRY.
2169         list-longnum =
2170             /sappsp/ci_numbers=>lookup( list-ebeln ).
2171     CATCH /sappsp/cx_number_not_found.
2172         *
2173         number does not exist.
2174     ENDTRY.
2175 ENDIF.
2176
2177 * Authorization for posting values company code level
2178 MOVE abap_false TO lv_flag_no_cc_auth.
2179
2180 READ TABLE organ WITH KEY werks = list-werks BINARY
2181 ENHANCEMENT-SECTION rm07docs_06 SPOTS es_rm07docs
2182 LOG-POINT ID /cwm/enh SUBKEY to_upper( sy-tcode &&
2183 * variable
2184 DATA: lv_fieldname TYPE fieldname.
2185
2186 * fieldsymbol
2187 FIELD-SYMBOLS: <field> TYPE any.
2188
2189 * check on general authorizations to display values a
2190 * general auth.check - display values and prices in 1
2191 CALL METHOD cl_sacf=>sacf_check_in_use
2192 EXPORTING
2193     id_name = 'MM_IM_VALUES'
2194 RECEIVING
2195     ed used = lv values chk.
```

Syntax error: missing closing parenthesis

ABAP | Ln 1225 Col 1 | NUM

Debugging ABAP Program

By

René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)

With the (F5 key) you can start debbuging your enhancement step by step



```
2162      CALL METHOD cl_change->is_active
2163      RECEIVING
2164          rv_active = lf_active.
2165      ENDIF.
2166
2167      IF lf_active = 'X'.
2168          TRY.
2169              list-longnum =
2170                  /sappsp/ro/cl_numbers=>lookup( list-ebeln ).
2171              CATCH /sappsp/ro/cx_number_not_found.
2172                  * number does not exist.
2173          ENDTRY.
2174      ENDIF.
2175
2176      * Authorization for posting values company code level
2177      MOVE abap_false          TO lv_flag_no_cc_auth.
2178
2179      READ TABLE organ WITH KEY werks = list-werks BINARY
2180      ENHANCEMENT-SECTION      rm07docs_06 SPOTS es_rm07docs
2181      LOG-POINT ID /cwm/enh SUBKEY to_upper( sy-tcode &&
2182      * variable
2183      DATA: lv_fieldname      TYPE fieldname.
2184
2185      * fieldsymbol
2186      FIELD-SYMBOLS: <field> TYPE any.
2187      * check on general authorizations to display values a
2188      * general auth.check - display values and prices in J
2189      CALL METHOD cl_sacf=>sacf_check_in_use
2190      EXPORTING
2191          id_name = 'MM_IM_VALUES'
2192      RECEIVING
2193          ed_used = lv_values_chk.
2194      IF NOT lv values chk IS INITIAL.
```


Debugging ABAP Program

By

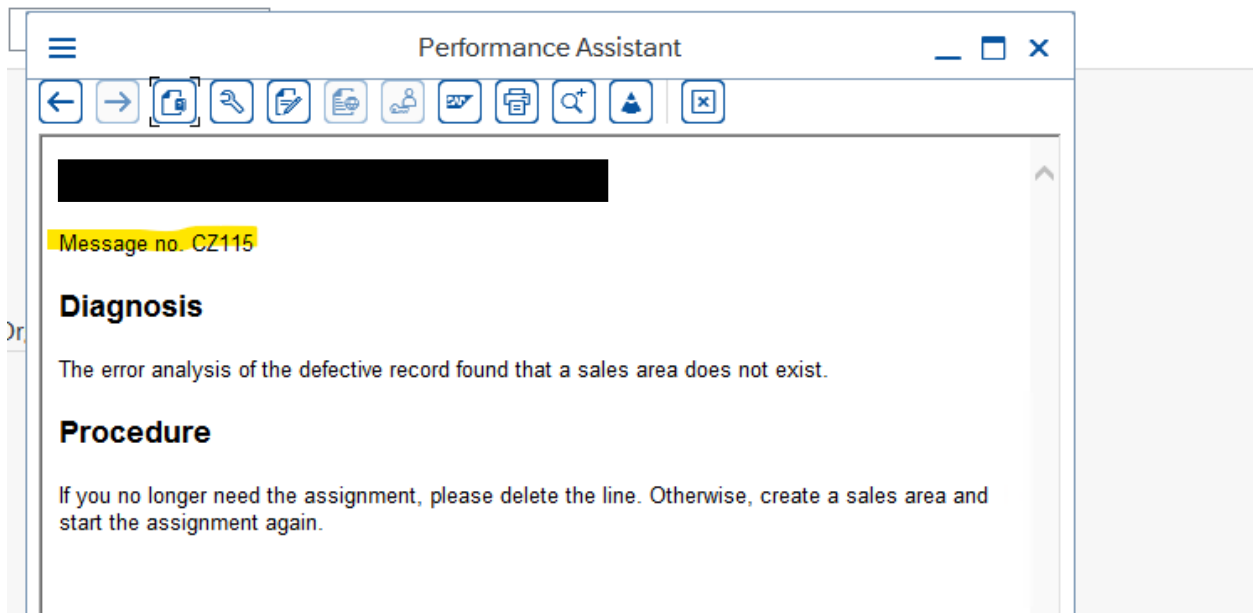
René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)

c) Debugging an error message

As I explained earlier, in some cases SAP error messages are not useful, which means that we don't know what caused the error. In such a situation we have no choice, the only method of troubleshooting is debugging.

Let's see how we can debug error message

A user sends you this screenshot by email:



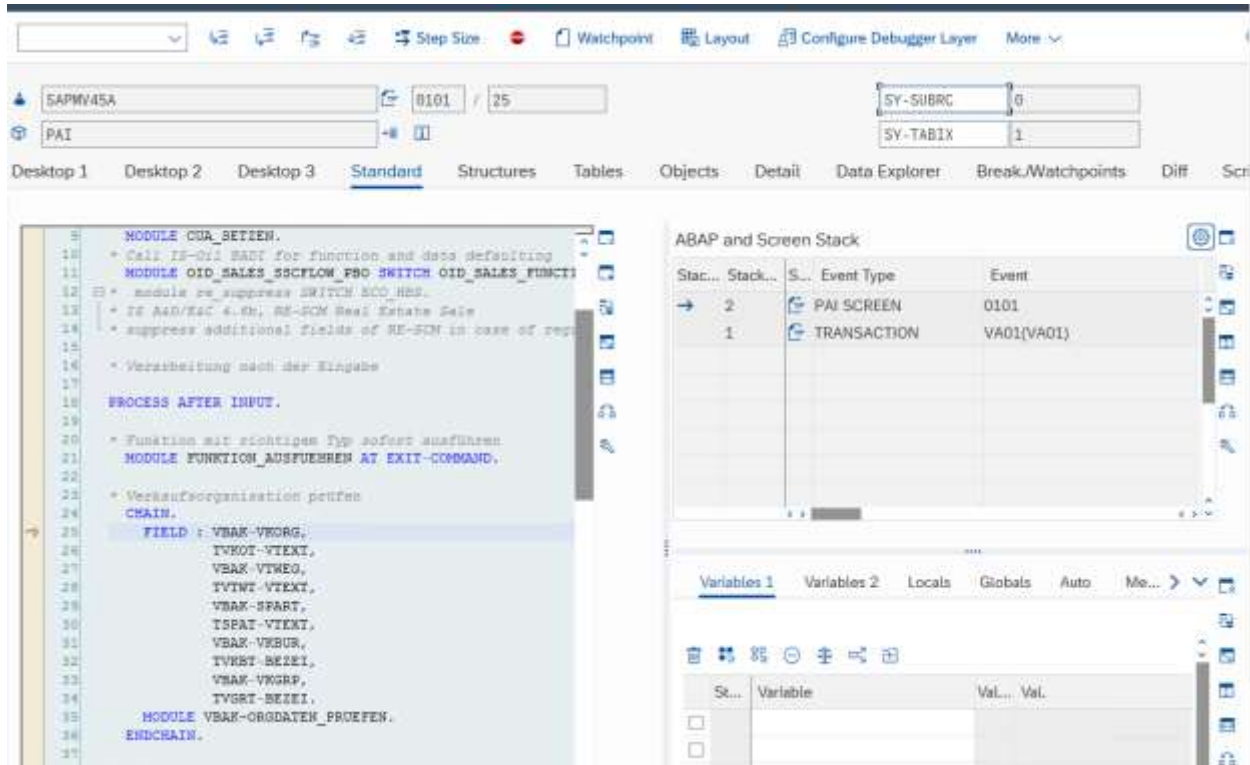
Our task now is to find out exactly which line of the ABAP code contains this error by debugging.

Open the program in debugging mode

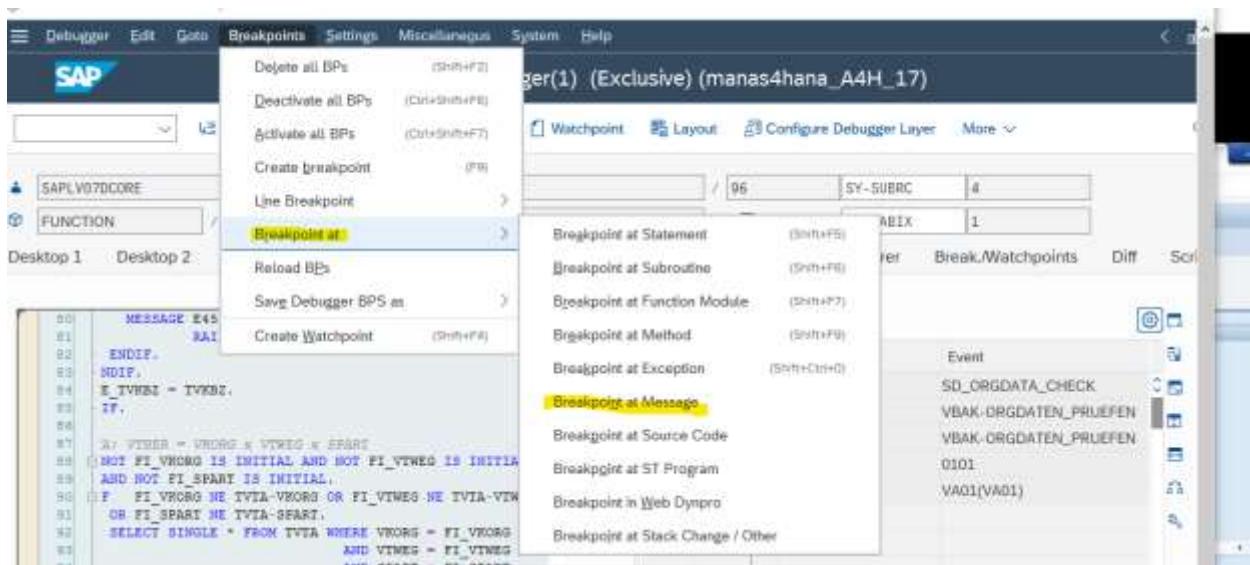
Debugging ABAP Program

By

René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)



Go to breakpoint at message

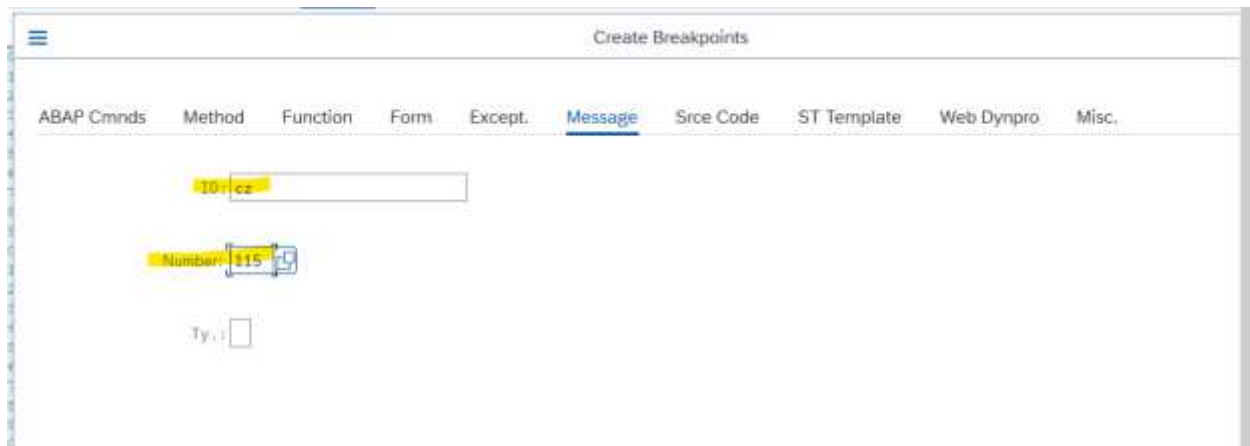


Debugging ABAP Program

By

René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)

Enter the information contained in the error message



Create Breakpoints

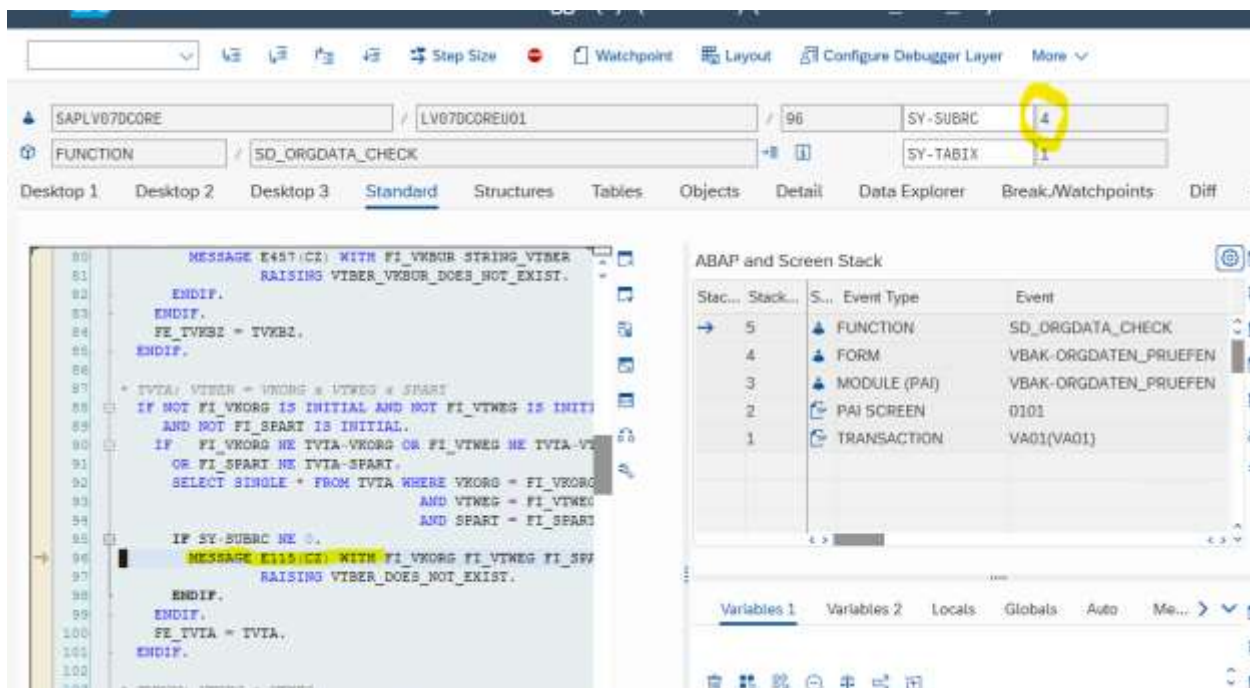
ABAP Cmds Method Function Form Except. **Message** Srce Code ST Template Web Dynpro Misc.

Message: E457(CZ)

Number: 115

Type:

And execute



SAPLV07DCORE / LV07DCOREU01 / 96 SY-SUBRC 4

FUNCTION / SD_ORGDATA_CHECK SY-TABIX 1

Desktop 1 Desktop 2 Desktop 3 **Standard** Structures Tables Objects Detail Data Explorer Break/Watchpoints Diff

```
90 MESSAGE E457(CZ) WITH FI_VKORG STRING VTBK
91 RAISING VTBK_VKORG_DOES_NOT_EXIST.
92
93 ENDIF.
94 FE_IVPSE = TVKBE.
95
96 ENDIF.
97
98 * IVTA: VTBK = VKORG & VTWEG & SPART
99 IF NOT FI_VKORG IS INITIAL AND NOT FI_VTWEG IS INITI
100 AND NOT FI_SPART IS INITIAL.
101 IF FI_VKORG NE IVTA-VKORG OR FI_VTWEG NE IVTA-V
102 OR FI_SPART NE IVTA-SPART.
103 SELECT SINGLE * FROM IVTA WHERE VKORG = FI_VKORG
104 AND VTWEG = FI_VTWEG
105 AND SPART = FI_SPART
106
107 IF SY-SUBRC NE 0.
108 MESSAGE E115(CZ) WITH FI_VKORG FI_VTWEG FI_SP
109 RAISING VTBK_DOES_NOT_EXIST.
110
111 ENDIF.
112 FE_IVTA = IVTA.
113
114 ENDIF.
```

ABAP and Screen Stack

Stack...	Stack...	S...	Event Type	Event
5	FUNCTION		SD_ORGDATA_CHECK	
4	FORM		VBAK-ORGDATEN_PRUEFEN	
3	MODULE (PAI)		VBAK-ORGDATEN_PRUEFEN	
2	PAI SCREEN		0101	
1	TRANSACTION		VA01(VA01)	

Variables 1 Variables 2 Locals Globals Auto Me... >

Debugging ABAP Program

By

René Rodrigue Efila Minkoulou (SAP S/4HANA MM/SD Consultant)

As you can see above the program has stopped exactly at the point where this error occurs.

Thank you