ABAP Debugger

The 2 debuggers

- Classic ABAP Debugger
- New ABAP Debugger

Frequently Asked Questions

- How do I debug running background processes or asynchronous processes?
- How do I debug completed background process?
- How do I debug Updates/System code?
- Why does it give a dump when I put a break-point in between SELECT and ENDSELECT? (classic debugger only)
- How do I enter the debugger (/H) from modal dialogs? (popup windows)
- How do I switch between the Classic and New Debugger?
- Why debugger don’t stop at my break-point?
- How to debug tRFC function module

How do I debug running background processes or asynchronous processes?

In transaction SM50 (process overview), you can select a background process and choose Program/Mode -> Program -> Debugging from the menu.

An alternative workaround, which allows you to step into a particular piece of code is to place a piece of code in a endless DO-ENDDO, where you can change a variable to step out of the DO at a particular point in your code. This allows you to hold the process at that point and debug it from SM50 as described above. An implementation of this exists in function module C160_HOLD_FOR_DEBUG, which will enter the endless loop if a particular environment variable is set, thereby allowing you to control its behaviour. (Further instructions are found in the comments in subroutine INC14Z_HOLD_FOR_DEBUG of include LC14ZFCB).

How do I debug completed background process?

You can do this only after the job has finished execution. This will simulate the exact background scenario with the same selection screen values as used in the job and sy-batch set to ‘X’.

- Use SM37 to get list of jobs, put the cursor on the job, type JDBG in the command line ( no ‘ ‘ ), and press ENTER
- You are in debug mode now. Step through SAP program (press F7 couple of times) until you get to code you need.
- Note: though running interactively, you can see that sy-batch = ‘X’.

How do I debug Updates/System code?

Both options are available from the menu in debugging. Choose Settings -> System/Update Debugging to activate either before proceeding.

Why does it give a dump when I put a break-point in between SELECT and ENDSELECT? (classic debugger only)

A breakpoint in SELECT loops can cause an exception through loss of the database cursor. The reason for this is that during debugging a database commit is triggered and hence the cursor is lost.

How do I enter the debugger (/H) from modal dialogs? (popup windows)

In modal dialogs, the okcode field where you usually enter /H is protected. You must use a SAPGUI shortcut, as explained in this wiki:

How do I switch between the Classic and New Debugger?

- From within the ABAP workbench, select the Utilities->Settings Menu
- Select the ABAP Editor Tab
- Select the Debugging tab within the ABAP Editor Tab
- Select the Classic Debugger or New Debugger radio button

Why debugger don’t stop at my break-point?

(or step by step, or step into, at a CALL statement, doesn’t go into the called procedure)
Many reasons:

<table>
<thead>
<tr>
<th>Context</th>
<th>Explanation</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>User exit (CMOD)</td>
<td>you try to debug a user exit (CMOD transaction)</td>
<td>Make sure you have activated both the function module AND the user exit</td>
</tr>
<tr>
<td>Update task</td>
<td>the code is called in an update task (CALL FUNCTION ... IN UPDATE TASK) and you did not activate the Update debugging and you didn't set the break-point after the update debugging has been started</td>
<td>enter /h and activate Update debugging, run the program. When the update debugging starts, press F9 key and enter the name of the function module to add the break-point. Note: a break-point added before the update debugging starts is ignored</td>
</tr>
</tbody>
</table>
| Asynchronous (aRFC, HTTP) | the code runs in asynchronous (aRFC, workflow) or in HTTP (BSP, WebDynpro, BI query, etc.) or is called from other systems or clients (RFC) | • Set an external break-point (see Note 688256 - Using HTTP/external debugging and Note 751872 - Problem analysis with HTTP or external debugging)  
  • External break-points are valid only 2 hours. Solution: add them again.  
  • As explained in note 868830 - Cannot debug an RFC in a HTTP session, if you are currently debugging HTTP, you can't go into a called RFC function module (even synchronous). The workaround is to set an external breakpoint in the called RFC function module, or switch to classic debugger (if the button is available). Note: you don't need an external breakpoint (step into works) if the ICF service option "GUI connection" is set to "Yes". |
| Background job           | it runs in background                                                       | SM37, select the job and use JDBG okcode, the job will starts interactively with sy-batch = 'X' (see Note 573128 - Debugging programs in the background) |
| tRFC                     | it runs in tRFC (CALL FUNCTION ... IN BACKGROUND TASK)                      | See How to debug tRFC in the FAQ below                                                      |
| run under different user or client | it runs under a different user, or in another client                      | you must both set break-points and run code within same client and with same user           |
| system program           | you set the break-point in a System program (program status "System") and you did not activate the System debugging | • enter /h and activate System debugging, run the program  
  • you may have to reenter /h and reactivate System debugging if break-point don't work again |
| program not restarted    | the program was already running and you didn't restart it and the Session Break-Point Immediately Active setting is off. This option is available in SE38, menu Utilities | Exit program and start it again                                                             |
|                          | Settings | ASAP Editor | Debugging                                           |                                                                                                      |
| Little bugs in enhancement framework | or you encountered one of the bugs since 7.0 (as I could see in SAPKB70013), because of some implicit enhancement options | enter /h and add a break-point at the beginning of a form, function module or method by using F9 key and entering its name |
| Little bugs in local test classes | or you added a break-point in a Local Test Class (AUnit). Debug doesn't work very well (as I could see in SAPKB70013) | A workaround is to call a procedure outside the test class and add a break-point in that procedure |
| no break because no call | the code is not called!                                                     | add a break-point at an upper level or use SE30 to trace calls to make sure the procedure is called |

How to debug tRFC function module

Say YYYY does CALL FUNCTION 'XXXX' IN BACKGROUND TASK ..., and you want to debug XXXX

- add breakpoint in YYYY before the CALL FUNCTION
- start YYYY
- when you reach the breakpoint, select menu Settings, Display/Change debugger settings, select the flag "tRFC (In Background Task): Block Sending". This flag means that, when the commit work occurs, the background task (FMs + parameters) is stored into database but not executed.
- run SM58
- Select the background task, and choose menu edit + debug LUW, the "background task" will start interactively and the debugger will stop at predefined points by sap, and in FM XXXX automatically.