

---

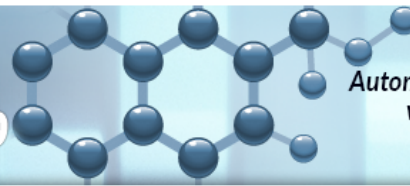
## **CT-Debug&Trace\_Module 4.0**

Add-On for SAP® ABAP™ Workbench

Automated testing and debugging  
of ABAP™ programs.

---

**SAP®** Certified  
Integration with SAP NetWeaver®



In future, the testing & debugging of the ABAP™ programs will become easier, simpler and more effective ...

Large scale single debugging - that's yesterday

## 1. Documentation of the processed commands and variables ...

The time consuming traditional way of debugging (F5, F6, F7 etc.) will be done in the future by your PC.

### Automatic documentation and trace recording

The screenshot displays the CT-Debug & Trace Module interface for program YWORK90\_BAS\_006. The main window shows the source code with line numbers 317 to 358. A 'Trace Control Panel' is overlaid on the code, showing recording status, current step (2490 of 2490), and control buttons like 'Continue recording', 'Single Step [F5]', 'Execute [F6]', 'Return [F7]', and 'Continue [F8]'. A variable table at the bottom left shows the current state of variables like SY-SUBRC, SY-TABIX, and WORK+OFF. A message window at the bottom shows system logs.

Name	Wert
SY-SUBRC	0
SY-TABIX	4
SY-DBCNT	27
<FS1>	Object t...
WORK+OFF (ZDFIES-L...	Object t...
WORK	11/SAPDMC/LSMW
ZDFIES-LENG	000073
OFF	28

### Screen Info's

- ① Each processed coding line is marked.
- ② 'Trace Control Panel' gives LIVE information on the actual debugging session recording process.
- ③ The variables of the current and previous source line are shown here.

- ④ Fast double-click navigation through the object tree.

You can now concentrate on the **really important coding lines**. So, leave the routine debugging to your PC. It is a lot quicker than you (much quicker, approx. 1000 - 1500 debugging steps/min.).

## 2. The manual debugging with automatic trace documentation ...

You can stop the **automatic debugging** and carry on with **manual debugging** and change back to the automatic debugging at **all times**. Because every machine or manual step is documented, an analysis of the previous debug steps is possible.

Additionally, all processed source code lines are marked, so you can see which line was executed and which not.

Optionally, the **synchronized** SE38 debugger screen (classic ABAP Debugger) is made available at the current coding place.

## 3. The Back trace mode - back stepping on the Frontend-PC ...

The CT-Debug & Trace\_Module makes an **offline-trace** available on the frontend PC. You are thus able to step forward & backwards through already **processed coding parts** and at the see the corresponding **contents of the processed variables** at that time.

Also **during a still active debugging session**, you are able to e.g. test a relevant coding part again.

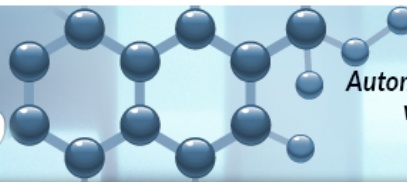
## Backtrace and variables contents...

The screenshot shows the CT-Debug & Trace Module interface. The main window displays the source code for the program 'ERZEUGEN-PSEUDOZUFALL'. A 'Trace-Liste für gewählte Zeile' dialog box is open, showing a list of 14 execution steps with their corresponding variable values. The 'Variablen' window shows the current state of variables like E\_RAND, Z\_A, Z\_P, and Z\_ZUFALL. The interface includes a menu bar, a toolbar, and a status bar at the bottom.

Index	StepNr.	Variablen
01	2213	Z_ZUFALL = 0.000000000..., Z_P= 9999999..., Z_A= 6713544... [...]
02	2227	Z_ZUFALL = 5.1778875707..., Z_P= 9999999..., Z_A= 6713544... [...]
03	2241	Z_ZUFALL = 1.9771160969..., Z_P= 9999999..., Z_A= 6713544... [...]
04	2264	Z_ZUFALL = 0.000000000..., Z_P= 9999999..., Z_A= 6713544... [...]
05	2278	Z_ZUFALL = 4.1794204547..., Z_P= 9999999..., Z_A= 6713544... [...]
06	2292	Z_ZUFALL = 8.7239914698..., Z_P= 9999999..., Z_A= 6713544... [...]
07	2306	Z_ZUFALL = 8.9024122104..., Z_P= 9999999..., Z_A= 6713544... [...]
08	2320	Z_ZUFALL = 6.7379423258..., Z_P= 9999999..., Z_A= 6713544... [...]
09	2343	Z_ZUFALL = 0.000000000..., Z_P= 9999999..., Z_A= 6713544... [...]
10	2361	Z_ZUFALL = 7.8112234740..., Z_P= 9999999..., Z_A= 6713544... [...]
11	2388	Z_ZUFALL = 0.000000000..., Z_P= 9999999..., Z_A= 6713544... [...]
12	2406	Z_ZUFALL = 4.0691552612..., Z_P= 9999999..., Z_A= 6713544... [...]
13	2424	Z_ZUFALL = 8.4537399312..., Z_P= 9999999..., Z_A= 6713544... [...]

### Screen Info's

- ① This ABAP™ source code line was processed 0014 times.
- ② Double-click on the '0014' will show you all of these 14 executions (step numbers) in a popup window.
- ③ Double-click on step number '2361' will show the variables at this point of time (see ④).
- ⑤ Through the navigation buttons '>' or '<' you can step forward/backwards in the recorded trace.



## 4. The variables are automatically determined and their content is documented ...

The variables of each processed coding lines are automatically determined. Then two snapshot of these variable are carried out: **Before and after** the execution of the coding line. So changes of the variable content are automatically recognized and also part of our trace documentation (max. 24 variables/command).

If you for example want to test if certain subroutines (**FORMs, FUNCTIONs, METHODs** etc.) were correctly executed, just have look at the trace documentation: All **subroutine parameters** (single fields, structures) **before and after** the call were saved.

## 5. Controlling the debugging session through debug profiles ...

The transcription of your debugging session will be supported considerably through **11 debug profiles**.

You control through a choice of a suitable profile the **details** which are recorded during the debugging session. You can for example let the CT-Debug&Tracer record each executed ABAP™ source lines (single step), or you can e.g. just record all subroutine, function module and method calls (with their parameters).

The program flow of large programs (e.g. transaction) is made transparent and important events and variable contents can be easier identified (compared to the classic).

You can also record only certain ABAP™ source code parts of a transaction (shown in our manual - section '**Tips&Tricks**'). An example would be the processing of only one given transaction screen or a special **user exit**. After reaching the desired source code include, just use one of our debug profiles to process it and document every step (with the contents of the variables).

## Debugging of large programs ...

The screenshot displays the CT-Debug & Trace Module interface for debugging program YWORK90\_BAS\_006. The main window shows a backtrace of the program, with the following code snippet visible:

```

879 E5_FIELD = D4_FIELD + 222.
880 ASSIGN CFELD+NFELD TO <GFS2>.
881
882 SKIP 3.
883 PERFORM SUBPRG-3 USING CFELD NFELD <FS1> <FS2> <GFS2>
884 CFELD+NFELD (IFELD).
885
886
887 ENDF0
    
```

The workspace tree on the left shows the program structure, including function modules like LSD03U02. The variable window at the bottom shows the current state of variables, with CFELD and NFELD highlighted. The call hierarchy window shows the current call stack, with the current call highlighted. The development class window shows the current development class, \$TMP.

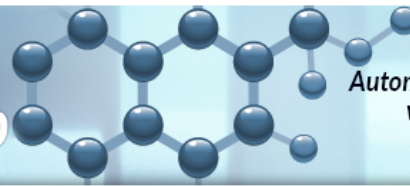
### Screen Info's

- ① The parameters of e.g. FORM/PERFORM are also recorded (step 1848).
- ② The parameter content is represented in the variable window (sizeable).
- ③ The interactive CALL hierarchy shows the position of the current FORM-Call (Step 1849) within the program flow.
- ④ The called function modules are shown in the workspace tree.
- ⑤ The current development class/package gives additional information.

## 6. The analysis mode with 20 interactive reports ...

Central component of the trace data analysis is the **integrated report engine** which was designed to handle large amounts of trace data. One important report is the debugging tree, which gives a quick hierarchical overview on the recorded debugging session (processed FORMS, FUNCTIONS, METHODS ...).

With a double click on a tree element, you can jump to more detailed data. In the **analysis mode** more than **20 interactive reports** are offered, and all reports are synchronized with the recorded debug session (cockpit information).



## 7. Type & coverage of the documentation of a debugging session ...

During an automatic debugging session of approx. **10 minutes**, you can e.g. automatically document more than **10,000 single program steps**, by using our single step trace profile. Additionally, approximately 20,000 variables with their contents (and content changes !!) are recorded in that time. No user activity is necessary here.

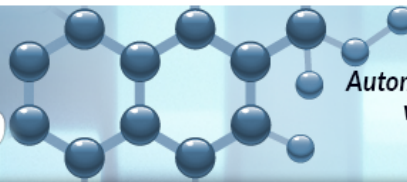
At the same **time all processed ABAP™ coding member** from the application server program library (e.g. **100 ABAP** includes, when tracing approx. **30,000 coding lines**) will be stored in our offline database for later **analysis, control and documentation purposes**. So you will get a consistent documentation of the program flow.

The processed **source code lines are marked in color** and additionally are synchronized with all our interactive trace reports.

## Extensive program flow documentation ...

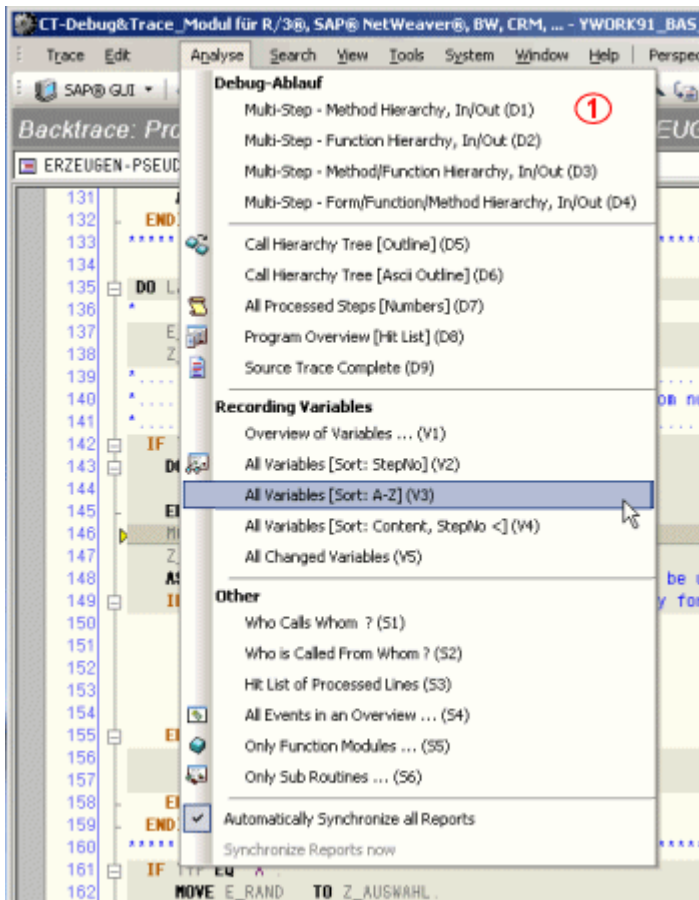
The screenshot displays the CT-Debug & Trace\_Module 4.0 interface. The main window shows the ABAP source code for program YWORK91\_BAS\_006, with lines 131-160 visible. The code includes logic for generating random numbers and handling data tables. The interface also features an 'Objekt-Tree' on the right, a 'Variablen' window showing current variable values, and two 'Trace' windows at the bottom. The first trace window shows a table of ABAP variables, and the second shows a table of ABAP variables sorted by name. The trace data includes step numbers, variable names, and their contents, such as Z\_ZUFALL and Z\_AUSWAHL.

PHMMSS.MS	STEPNO	LINENO	C	P	VARIABLE_NAME	VARIABLE_CONTENT
121522967	0002365	000145			SY-DBCNT	1
121522967	0002365	000145			SY-SUBRC	0
121522967	0002365	000145			SY-TABIX	1
121522967	0002365	000145	X	P	Z_ZUFALL	5.3856137362226448E+00
121523030	0002366	000143			LAENGE	2
121523030	0002366	000143			SY-DBCNT	1
121523030	0002366	000143			SY-SUBRC	0
121523030	0002366	000143			SY-TABIX	1
121523264	0002367	000146		P	LAENGE	2
121523264	0002367	000146			SY-DBCNT	1
121523264	0002367	000146			SY-SUBRC	0
121523264	0002367	000146			SY-TABIX	1
121523264	0002367	000146			Z_AUSWAHL	00000000000000000000000000000044
121523264	0002367	000146			Z_ZUFALL	5.3856137362226448E+01
121523374	0002368	000147			LAENGE	2
121523374	0002368	000147			SY-DBCNT	1
121523374	0002368	000147			SY-SUBRC	0
121523374	0002368	000147			SY-TABIX	1
121523374	0002368	000147	X	P	Z_AUSWAHL	00000000000000000000000000000054
121523374	0002368	000147			Z_OFFSET	20



## Screen Info's

- ① 20 interactive reports are available.
- ② The report 'V3' displays a list of all recorded variables. So you can immediately have a look at the variable 'Z\_ZUFALL' at different processing times.
- ③ If you double click on Step No. '02367' all other reports are automatically synchronized, so each report show the same step number (which corresponds to a time code).



## 8. The intensive and thorough testing of programs ...

Through the **auto-debugging** feature (with its control through debug profiles), you can process your program and **extensively** test it in a short time. The automatic documentation of each debugging step can later be added to the technical program documentation as a **test protocol**.

It is therefore possible, at a later time, to browse **forward and backwards** in the **offline-mode**. You will have an interactive documentation of your program flow **at a specific time**, which can be (re)processed at any time in the future (without any connection to the application server). So if an ABAP™ source is changed or deleted in the future, the trace documentation will stay consistent.

It is therefore possible at a later time, to browse with the essential trace recordings, **forward and backwards** in the **offline-mode** and **at the same time** analyse the variables that were involved. The current contents of the **processed variables** are available over the whole course of the program.



## 9. Separating documentation from analysis in debugging ...

The possibility to automatically record the complete program flow in a short time, leads to a more efficient course of work with the **testing of programs** and the **solving of bugs**:

- To start with, the debugging session is **documented** (your PC does this for you in the background)
- This followed by an **analysis** which is supported by the **dedicated reporting features**.

Also consideration has to be given, that the complete program flow is automatically part of the documentation, which includes **access to control, master und other data tables and the used test data**.

You can then process the exact recorded program flow at any time with our offline trace feature. In contrast: It is often quite difficult to reproduce the exact program flow manually (e.g. database tables were updated during the last test run).

## 10. Further Add-On debugging functions ...

- **LIVE** information on the actual debugging session recording process through the '**Trace Control Panel**',
- Ability to go back to already processed program lines **at any time** (back trace feature), even after a short dump
- **Code folding feature** (outlining) makes it easy to understand large programs (e.g. a large CASE statement can be folded and unfolded directly in the debug session,
- **Trace pool** for the reconstruction of, test data, the program flow, situations where errors occurred,
- **Syntax highlighting** of ABAP™ sources (report, program, function module, classes ...),
- Object browser (programs, classes, function groups, structures ...) can e.g. search for certain variables,
- Work space tree of the **actual processed** source code member, a history of the session,
- **Multi search functions** with regular expressions, the result can be displayed in multiple windows,
- Display of **tool tips** when placing the mouse cursor on certain coding parts (e.g. declaration data of variables is displayed),
- Modern eclipse like user interface, each GUI element can be individually arranged, these window arrangements can be saved and restored as a **perspective** ...

## 11. Communication with SAP® R/3®, mySAP(™) ERP, SAP Netweaver(™) ...

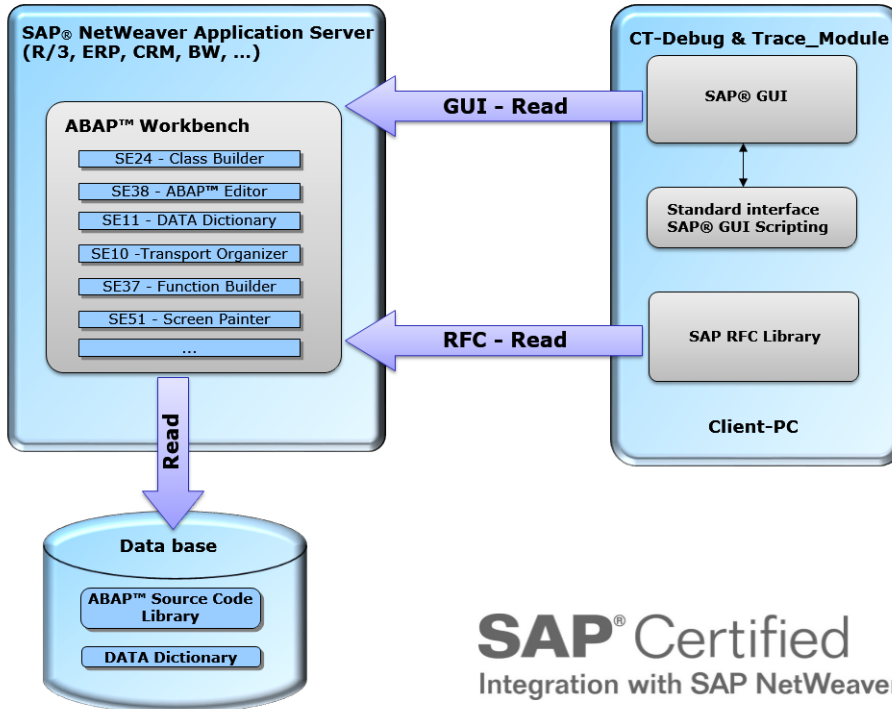
The CT Add-On is suitable for the following SAP® systems: R/3® from release 3.1x up to 4.7 (Enterprise), mySAP™ ERP, mySAP™ CRM, SAP® BW, SAP® APO and SAP Netweaver™ (up to 6.4).

The CT-Debug & Trace\_Module uses a modern GUI library, which can be fully customized. More than 20 GUI elements (display windows, trees, control elements ...) can be individually arranged to build your individual environment.

## Diagram

### Communication between SAP® NetWeaver Application Server and CT-Debug & Trace\_Module

The CT-Debug & Trace\_Module communicates via **Read-only** with the application server:



SAP® Certified  
Integration with SAP NetWeaver®

## 12. Contact

CT-Softwareberatungs GmbH  
Ziegeleiweg 8  
33415 Verl  
Germany

Web: [www.ct-software.com](http://www.ct-software.com)  
Email: [sales@ct-software.com](mailto:sales@ct-software.com)  
Tel: +49-(0)5246-9310-15  
Fax: +49-(0)5246-9310-16

Copyright © 2013 CT-Softwareberatung GmbH. All rights reserved. Microsoft®, WINDOWS®, NT®, XP® are registered trademarks of Microsoft corporation. SAP®, R/3®, mySAP™, ABAP™ are trademarks of SAP AG. All other products mentioned are the trademarks of the respective companies.