



The New ABAP Debugger (NetWeaver 2004s)

Christoph Stoeck
Boris Gebhardt

THE BEST-RUN BUSINESSES RUN SAP® 



- **Motivation**
- **Architecture & Handling**
- **UI Basics**
- **New ABAP Frontend Editor**
- **Variable Display & Navigation**
- **Comparing Variables (Diff-Tool)**
- **Breakpoints & Watchpoints**



- **Motivation**
- Architecture & Handling
- UI Basics
- New ABAP Frontend Editor
- Variable Display & Navigation
- Comparing Variables (Diff-Tool)
- Breakpoints & Watchpoints

Motivation – The Classic Debugger

The screenshot displays the SAP ABAP Debugger interface. At the top, there is a menu bar with options: Debugging, Edit, Goto, Breakpoints, Settings, Development, System, and Help. Below the menu is a toolbar with various icons. The main window is titled "ABAP Debugger" and contains several tabs: Fields, Table, Breakpoints, Watchpoints, Calls, Overview, and Settings. The "Table" tab is active, showing a table with one row: "nodetab []" with content "Table [6x1184]". Below the table, there are fields for "SY-SUBRC 0", "SY-TABIX 1", "SY-DBCNT 1", and "SY-DYNNR 0100". A status bar at the bottom indicates "Field copied into the display".

```
FUNCTION WB_TREE_RETURN_OBJECT_LIST
  g_nodetab-force_plus = 'X'.
ENDIF.
APPEND g_nodetab TO nodetab.
ENDWHILE.
ENDLOOP.
ELSE.
  * table EXPAND is initial; pass on the complete object list:
  nodetab[] = g_nodetab[].
ENDIF.

IF treename(3) NE 'IA_' " (no special preparations for list
AND treename(3) NE 'WY_' " (no special preparations for Web
```

We have already a powerful
ABAP debugger.

Why do we need a new
one ??



Motivation – The New ABAP Debugger

The screenshot displays the SAP ABAP Debugger interface. The main window shows the source code of a function module. A callout bubble points to a specific line of code. To the right, there are panels for 'Table Contents' and 'Local Variables and Parameters'.

Code snippet (lines 440-451):

```
440 WHILE g_nodetab-next > 0.  
441   READ TABLE g_nodetab WITH KEY g_nodetab-n  
442   IF sy-subrc NE 0. EXIT. ENDIF.  
443   IF g_nodetab-child > 0.  
444     g_nodetab-force_plus = 'X'.  
445   ENDIF.  
446   APPEND g_nodetab TO nodetab.  
447   ENDWHILE.  
448   ENDLOOP.  
449 ELSE. VALUE = Standard Table[6x124(1104)]  
450   * tab] TYPE = Standard Table[6x124(1104)] the complete  
451   nodetab[] = g_nodetab[].  
452 ENDIF.
```

Table Contents (NODETAB []):

Line	ID[N(6)]	TYPE[C(4)]	NAME[C(30)]	TLEVEL[N(2)]
1	000056	OFM	CURRENT_START_03	
2	000057	OFM	GET_CHANGED_VALI03	
3	000058	OFM	GET_FOCUS_03	
4	000059	OFM	LOAD_TABLE_DATA_03	
5	000060	OFM	SET_DYNP_VALUES_03	

Local Variables and Parameters:

V. St.	Variable Name	V. Val.	Technical
	TRENAME	P6_SAPLSTPDA_TOOL_TABLE_V C(45)	
	WITH_ROOT	X	C(1)
	REFRESH		C(1)
	HIDDEN_EXPAND		C(1)
	BROWSER_MODE	REP	C(3)
	FILTER_MODE		C(70)
	NODETAB	000000	Flat Struct
	EXPAND	Structure: flat & char11k	Flat Struct
	EXPAND_ID	000055	N(6)

... Therefore

- The first version of the New ABAP Debugger is available with NetWeaver04. The full range of features described in the presentation is available with NetWeaver 2004s.

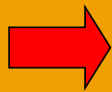
Classic Debugger

Technology

- Debugger and debuggee run in the same (internal) session
- Debugger dynpros placed “in-between”

Consequences

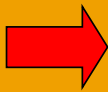
- Not all ABAP code can be debugged (no RPERFs: Conversion / Field exit))
- Not free of side effects (F1, F4 help, list output)
- Implementation of new features not always straight-forward
- No chance to use modern UI techniques (no ABAP allowed in the debugger !)



We need a new ABAP debugger technology

**Higher productivity for development & support
using ABAP debugger**

- **More robust debugger architecture (no side effects)**
- **Possibility to implement new features (e.g. a diff tool for internal tables) faster and with less risks**
- **More flexible & extensible state-of-the-art debugger UI**

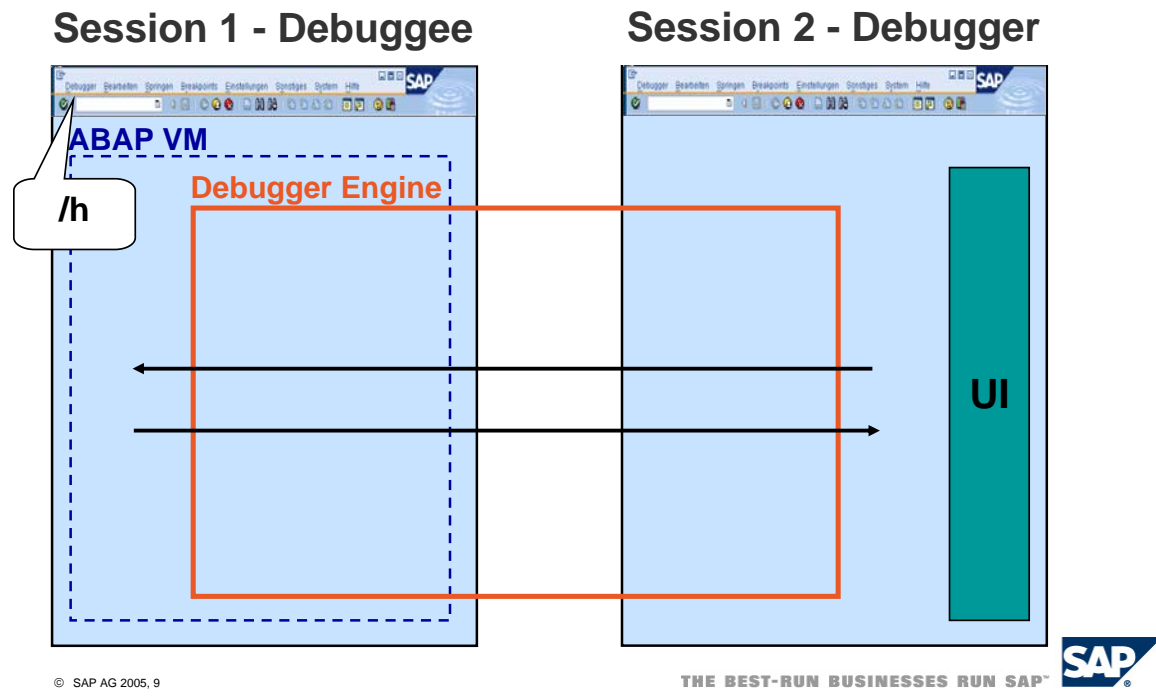


**Use two separated sessions for the
debugger and the application**



- Motivation
- **Architecture & Handling**
- UI Basics
- New ABAP Frontend Editor
- Variable Display & Navigation
- Comparing Variables (Diff-Tool)
- Breakpoints & Watchpoints

The New Debugger is attached to an “external session”



The classic Debugger runs in the same internal session as the application itself. Therefore after entering a new roll area (submit/call transaction) the debugger is restarted-> all debugger breakpoints are gone and the UI is refreshed. (You have to type in your variables again...).

The new ABAP Debugger is totally separated from the application and is attached to the external session and not the internal session. Therefore you can debug roll area (internal session) switches without any disturbance.

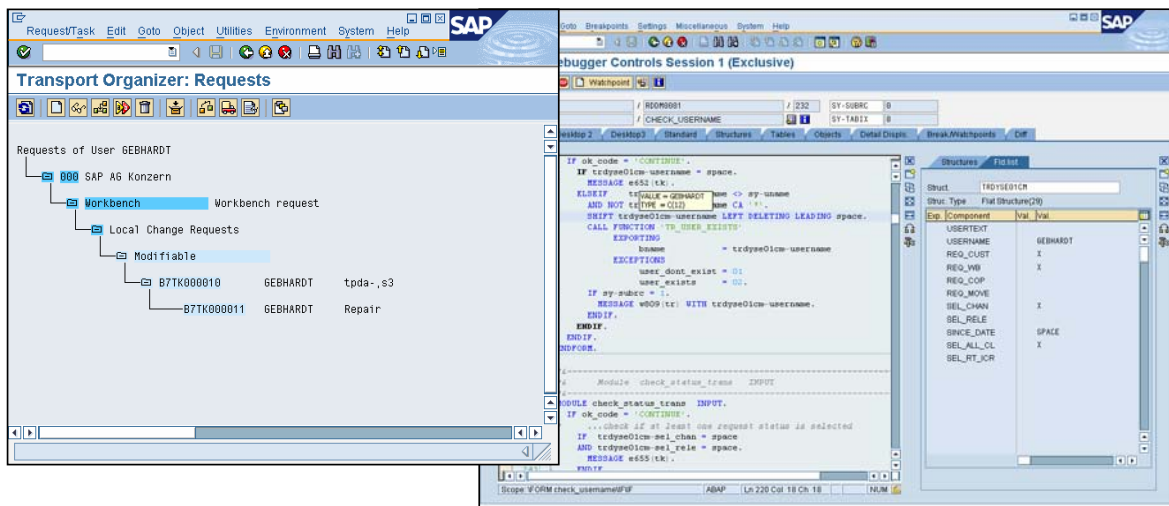
Two Process Architecture - Consequences

The screenshot displays two SAP windows side-by-side. On the left is the 'Transport Organizer' window, showing a list of requests for user 'GEBHARDT'. On the right is the 'ABAP Debugger Controls Session 1 (Exclusive)' window, showing the source code of the 'check_username' function. A callout bubble with the text '/h' is positioned above the Transport Organizer window. A yellow arrow points from the Transport Organizer window to the ABAP Debugger window.

- The New Debugger is started in a separated session, after prompting “/h”
- The debuggee is inactive while the debugger is active.

Advantage: During debugging you still see your last screen input

Two Process Architecture - Consequences



- The debugger is still available but inactive, when the program finished
- The debugger is not closed as long as the debuggee session is alive !
- You may detach the debugger by prompting “/hx” in the debuggee session

Advantage: The debugger with all your settings, variables, breakpoints,... is always available, when you restart debugging !

© SAP AG 2005, 11

THE BEST-RUN BUSINESSES RUN SAP™



- As the New ABAP Debugger is attached to external session the debugger is not closed, when the application finished.
- Only when you close the screen or if you use ok-code /hx the debugger is detached.
- You may use the debugger setting (Menu: Setting ->Display/Change debugger Settings): “Close Debugger After 'F8' and Roll Area End “ to influence this behavior.
If you switch on this setting, then the Debugger will be closed automatically after pressing “Continue” (F8) and the end of the roll area. This is more or less the behavior of the Classic Debugger, and in order to make the migration smooth this is the default setting.

Both debugger: differences in handling

	Classic Debugger	New Debugger
Subject that is being debugged	<p>Internal session (roll area)</p> <ul style="list-style-type: none"> ■ (breakpoints) ■ UI settings ■ tool content, e.g. variables <p><u>lost</u> in new roll area</p>	<p>External session (GUI window)</p> <ul style="list-style-type: none"> ■ breakpoints ■ UI settings ■ tool content, e.g. variables <p>NOT lost</p> <p>-> /hx closes debugger</p> <p>-> option: close automatically</p>
Window where the debugger is presented in	<p>Same window like application</p>	<p>Separate window</p> <ul style="list-style-type: none"> ■ last input screen visible ■ focus follows active window



- Motivation
- Architecture & Handling
- **UI Basics**
- New ABAP Frontend Editor
- Variable Display & Navigation
- Comparing Variables (Diff-Tool)
- Breakpoints & Watchpoints

New Debugger UI – Main Parts

The screenshot displays the SAP ABAP Debugger interface with several key components highlighted by callouts:

- Control Area:** Located at the top left, it contains buttons for 'Watchpoint' and 'Break/Watchpoints'.
- Desktops:** A row of tabs below the Control Area, including 'Desktop 1', 'Desktop 2', 'Desktop 3', 'Standard', 'Structures', 'Tables', 'Objects', 'Detail Display', 'Break/Watchpoints', and 'Diff'.
- Process Info:** A box at the top right showing session details like '(1) - ABAP Debugger Controls Session 1 (Exclusive)'. Below it, fields for 'SY-SUBRC' and 'SY-TABIX' are visible.
- Source line / SY-Fields:** A box pointing to the main code editor window, which shows ABAP source code with line numbers (12-37) and a current execution point at line 20.
- Tools:** A box pointing to the 'Scope: METHOD handle_expand_no_children' and the 'ABAP and Screen Stack' table below the code editor.
- Data Objects:** A window on the right titled 'Data Objects' showing a list of objects with columns for Name, RE, Type, and Value. Objects include EVENTID_LINK_C, ITEM_CLASS_BU, etc.

© SAP AG 2005, 14

THE BEST-RUN BUSINESSES RUN SAP®

■ The process info contains:

- ◆ The session number you are attached to
- ◆ The debug mode (/hs-> system debugging active/ RFC / HTTP/ATTACHED etc.)
- ◆ Info: Is this an exclusive or non-exclusive debug session
 - In a non-exclusive debug session each Rollin/Rollout e.g. during a debug step is connected to a db commit
 - In an exclusive debug session the current work process is locked for your debug session, therefore there is no need for a db commit during rollout/rollin, because you will always re-enter the same locked work process.
 - The system always tries to get an exclusive debug session when you start debugging (in a productive client only exclusive debug sessions are allowed!)
 - But the number of exclusive debug session is limited by the profile parameter rdisp/wpdebug_max_no, because if you would lock all dialog work processes with an exclusive debug session then nobody would be able to logon ...
 - In a development system we recommend rdisp/wpdebug_max_no = number of dialog processes / 2.
 - All this is valid for the Classis and the New ABAP Debugger !

New Debugger UI - Desktops

User specific desktops

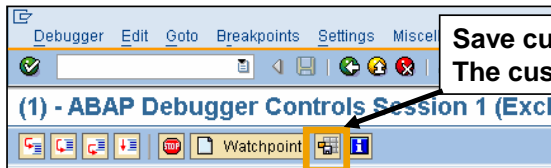
Standard desktops



The New Debugger provides:

- Three user specific desktops, which you can customize and save as your favorite debugger environment
- Seven standard desktops, which should cover most of the common working conditions in the debugger:

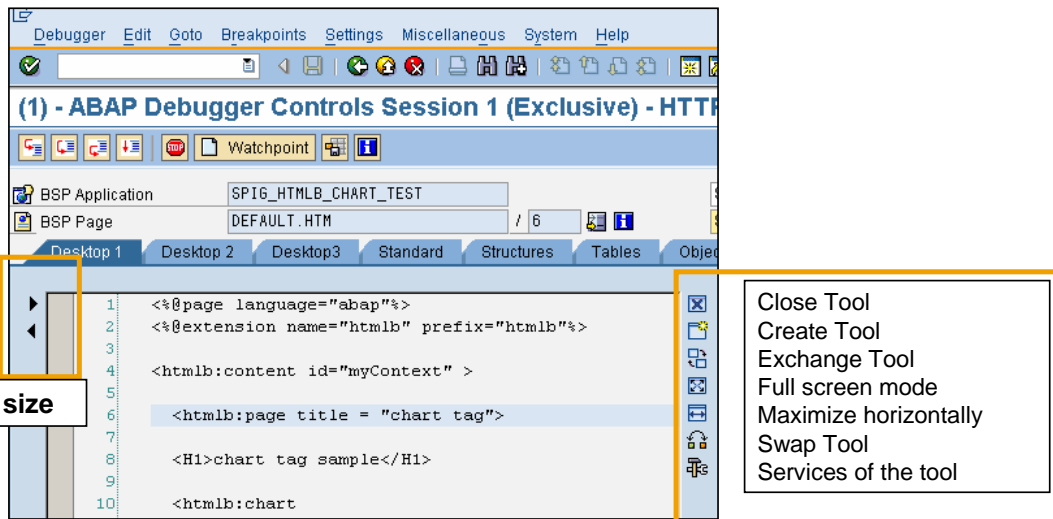
- | | |
|------------------------------|---------------------------------------------------------------|
| • Standard: | Stepping through the code (Editor, Stack, Quick Watch) |
| • Structures: | Display structures |
| • Tables: | Display tables |
| • Objects: | Display objects |
| • DetailDispls: | Display strings , simple fields ... |
| • Break-/Watchpoints: | Maintain your break-/watch-/checkpoints |
| • Diff | Compare Variables |



**Save current layout of the user specific desktops.
The customizing of the standard desktops is NOT saved !**

Customize The New Debugger UI – Customize layout

With the standard “Back” button (F3) you can “Undo” all your layout changes



Layout customizing can be also done via context menu.

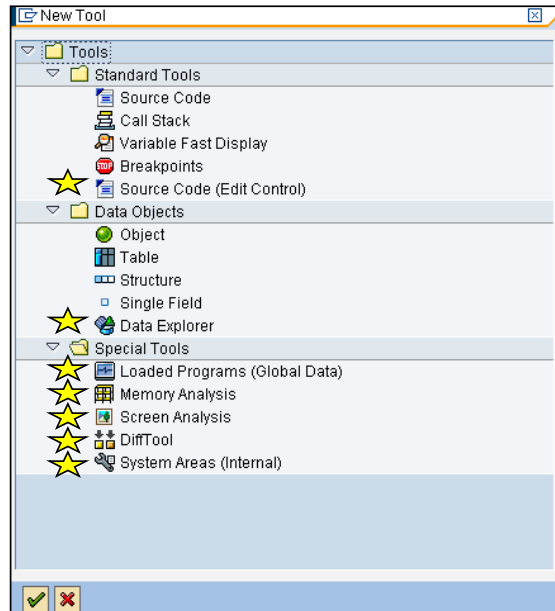
© SAP AG 2005, 16

THE BEST-RUN BUSINESSES RUN SAP® 

- The New ABAP Debugger (version NetWeaver 2004s) allows debugging of HTTP sessions as well. In this screenshot you see the debugging of a Business Server Page (BSP).

Customize The New Debugger UI – Tools

Available Tools:
 ★ **New with NetWeaver 2004s**



You may create up to 4 instances of one tool on one desktop. (e.g. to compare 4 internal tables)

© SAP AG 2005, 17

THE BEST-RUN BUSINESSES RUN SAP™



■ Standard Debugger Tools:

- ◆ (Tools, every debugger should have)
- ◆ Source Code Display (Current source code extract)
- ◆ Stack Display (Current ABAP & Screen stack)
- ◆ Variable Fast Display (Quick watch of variables)
- ◆ Breakpoint Display (Maintain your breakpoints)

■ Detail displays for variables:

(Tools, which are specialized for the analysis of one dedicated ABAP data type)

- ◆ Internal Table Display (Internal Tables)
- ◆ Object Display (Objects and classes)
- ◆ Structure Display (Structures)
- ◆ Single Field Detail Display (Simple data types, like C, N, D, T, STRING, ...)
- ◆ Data Explorer variable display (Generic and tree like)

■ Special tools:

- ◆ Diff Tool (Compare variables)
- ◆ Memory Inspector (Integrated Memory Inspector^[1])
- ◆ Loaded Programs (Attributes of all loaded programs and their

Customize The New Debugger UI – Tool Services

All features of a tool can be accessed via the tool service button...

The screenshot shows the SAP debugger interface with the 'Table Services' menu open. The main window displays a table with columns 'Line', 'MANDT(C(3))', 'CARRID(C(3))', and 'CONNID(N)'. The 'Table Services' menu is open, showing options like 'Save to Local File', 'Search', 'Find Next', 'Edit Columns', 'Save Columns', 'Change Row Content', 'Insert Row (Index)', 'Append Row (APPEND)', 'Delete Rows', and 'Delete Table (FREE)'. A context menu is also shown on the right, listing the same options.

...or the context menu

© SAP AG 2005, 18

THE BEST-RUN BUSINESSES RUN SAP®



- In order to make it easy finding all the features of the different tools, each tool uses a menu tree (the so called tool services) .
- The tool services are separated in standard services (download, search) which are provided by more or less all tools and the tool specific services (e.g. for the table view: Change row content)
- By the way, in the New ABAP Debugger you can change the content of internal tables in a very convenient way:
 - ◆ Insert/Append using a template (e.g. insert a copy of table line 11 and I will do only some changes before the real insert)
 - ◆ Free of an internal table



- Motivation
- Architecture & Handling
- New ABAP Debugger UI Basics
- **New ABAP Frontend Editor**
- Variable Display & Navigation
- Comparing Variables (Diff-Tool)
- Breakpoints & Watchpoints

```
1  METHOD handle_evt_tree_double_click .
2
3
4  DATA:
5    l_selected_object  TYPE repitem,
6    l_function         TYPE sy-ucomm.
7
8  CHECK node_key <> 'TREE' AND
9    NOT node_key IS INITIAL.
10
11 * get node data VALUE = 000000000027PO STATUS_0100
12 READ TABLE me->tTYPE = Flat Structure(848)
13 INTO l_selected_object
14 WITH KEY node_key = node_key.
15
16
17 * -----
18 * no functions for root node of inactive objects list
19 CHECK NOT ( me->active_tree->root_object_type EQ 1 ) .
20 AND l_selected_object-node_key EQ 1 ) .
21 * no functions for certain other object types:
22 CHECK NOT l_selected_object-objtype = swbm_c_type_c
23 * CHECK NOT ( me->browser_mode = swbm_c_type_wdy_
24 * AND l_selected_object-objtype = swbm_c_
25 * -----
26
```

Main features

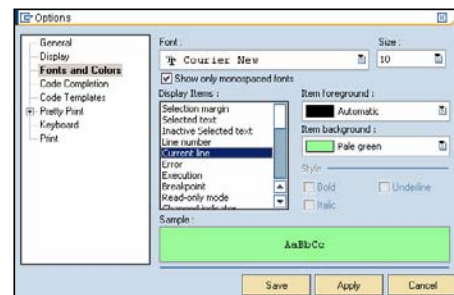
- Syntax coloring
- Block marking / folding
- Display variable content & type
- Easy BP handling

New ABAP Frontend Editor is available with NetWeaver 2004s plus 6.40 GUI (PL10).

- Please check SDN (<https://www.sdn.sap.com>) for a detailed descript of the New ABAP Editor (“**Inside the New ABAP Editor**”)

```
440 WHILE g_nodetab-next > 0.  
441   READ TABLE g_nodetab WITH KEY g_nodetab-next  
442   IF sy-subrc NE 0. EXIT. ENDIF.  
443   IF g_nodetab-child > 0.  
444     g_nodetab-force_plus = 'X'.  
445   ENDIF.  
446   APPEND g_nodetab TO nodetab.  
447   ENDWHILE.  
448   ENDOLOOP.  
449 ELSE.  
450   * table EXPAND is initial; pass on the complete ob:  
451   nodetab[] = g_nodetab[].  
452 ENDIF.  
453  
454  
455 IF treename(3) NE 'IA_' " (no special p  
456 AND treename(3) NE 'WY_' " (no special p  
457 * enter the status (active/inactive) of the node o  
458 PERFORM prepare_for_display TABLES nodetab  
459 USING treename.  
460 ENDIF.  
461  
462  
463  
464 * significant only for persistent classes:  
465 IF l_idexp IS NOT INITIAL.
```

- Syntax coloring
- Block marking / folding
- Display scope
- Convenient vertical and horizontal scrolling
- Powerful search functionality
- Control (layout, colors, ...) can be customized



ABAP Frontend Editor – Display Variable Content

```
1 METHOD handle_evt_tree_double_click .
2
3
4 DATA:
5   l_selected_object TYPE repitem,
6   l_function        TYPE sy-ucomm.
7
8 CHECK node_key <> 'TREE' AND
9       NOT node_key IS INITIAL.
10
11 * get node data VALUE = 000000000027PO STATUS_0100
12 READ TABLE me->t TYPE = Flat-Structure(846)
13 INTO l_selected_object
14 WITH KEY node_key = node_key.
15
16
17 -----
18 * no functions for root node of inactive objects
19 CHECK NOT ( me->active_tree->root_object_type
20            AND l_selected_object-node_key EQ 1 )
21 * no functions for certain other object types:
22 CHECK NOT l_selected_object-objtype = swbm_c_type
23 * CHECK NOT ( me->browser_mode = swbm_c_type AND
24             AND l_selected_object-objtype = swbm_c_type )
25 *
26
```

Mouse pointer hovers over variable:
-> Variable content and type is displayed in a quick info window.

The 'Options' dialog box is shown with the 'Display' section expanded. The following options are visible:

- Enable Current Scope
- Enable QuickInfo
- Show tooltip on hovering, after: 500 msec
- Enable Code Hints
- Display know completions when they save: 2 chars
- AutoHide tooltips after: 3000 msec

Buttons at the bottom: Save, Apply, Cancel.

In the control option dialog you can disable the quick info or adapt the reaction time.

ABAP Frontend Editor – Set Breakpoints

```
1  □ METHOD handle_evt_tree_double_click .
2
3
4  DATA:
5      l_selected_object TYPE repitem,
6      l_function        TYPE sy-ucomm.
7
8
9      CHECK node_key <> 'TREE' AND
10         NOT node_key IS INITIAL.
11
12 * get node data
13 READ TABLE me->tree_node_data
14         INTO l_selected_object
15         WITH KEY node_key = node_key.
16
17 □ * -----
18 * no functions for root node of inactive objects list
19 CHECK NOT ( me->active_tree->root_object_type EQ 1
20             AND l_selected_object-node_key EQ 1 ) .
21 * no functions for certain other object types:
22 CHECK NOT l_selected_object-objtype = swbm_c_type_c
23 □ * CHECK NOT ( me->browser_mode = swbm_c_type_wdy_
24             AND l_selected_object-objtype = swbm_c_
25 * -----
26
```

scope:METHOD handle_evt_tree_do... ABAP Ln 10 Col 1 Ch 1 NUM

Set / Delete breakpoints by clicking in the left (breakpoint) column .
Double-Clicking in the code will no longer set or delete breakpoints.
Deactivate /Activate the breakpoint via context menu.

ABAP Frontend Editor – Context Menu

```
1
2 METHOD handle_expand_no_children.
3   DATA: c_nodes TYPE treem_c_nodes, " nodes which must be
4         " copied to the view
5         c_node TYPE REF TO cl_tree_model_node,
6         view_key_to_object_entry TYPE treem_view_key_to_object,
7         node TYPE REF TO cl_tree_model_node,
8         child_node TYPE REF TO cl_tree_model_node,
9         copied_nodes TYPE i,
10        queue TYPE STANDARD TABLE OF treem_c_node,
11        sum TYPE i,
12        max_node_copy_count type i.
13
14 CALL METHOD base_tree_control->collapse_subtree
15
16 * get
17 READ
18 WI
19 IN
20 IF
21 CA
22 ENDI
23 node
24
25 * chec
26 IF N
27 IF
28 *
29 *
30 *
31
32 EN
33 CL
34 CL
35 * c2
36 CALL METHOD base_tree_control->collapse_subtree
37 EXPORTING
```

- Editor Context Menu contains:**
- local control features like "Find", "Outlining"
 - Debug features like maintain BP, Jump to statement ...)
 - Other services including layout customizing



- Motivation
- Architecture & Handling
- UI Basics
- New ABAP Frontend Editor
- **Variable Display & Navigation**
- Comparing Variables (Diff-Tool)
- Breakpoints & Watchpoints

Variable Display I

Quick info in ABAP Front-end Control

```
34 CLEAR node->ex VALUE = {0:110*\CLASS=CL_GUI_COLUMN_TREE}
35 * close folder TYPE = Reference
36 CALL METHOD base_tree_control->collapse_subtree
37 EXPORTING
```

Variable Fast Display

St	Variable	V	Val.	C	Hexadecimal Value	Technical Type	Absolute Type	Read-Only
	BASE_TREE_CONTROL		{0:110*\CLASS=CL_GUI_COLUMN_TREE}			Reference	\TYPE=%_T00004S	<input type="checkbox"/>

Detail View

Sta	Bin	Visi	Attrib.	Val.	Val.	Ch.	Hexadecimal Value
			OBJECT				
			CL_GUI_OBJECT				
			ACTIVEX		X		58
			CATT_ACTIV				20
			GUI_IS_RUNNING		X		58

© SAP AG 2005, 26

THE BEST-RUN BUSINESSES RUN SAP



- The New ABAP Debugger allows analyzing variables on different detail level:
 1. While stepping through the code you can just use the quick info to get the value and the type
 2. A double-click on the variable name leads to the variable fast display where you get some extra info (e.g. the hex value, absolute type (how is the variable declared), is the variable read only ?
 3. A double-click on the variable name in the variable fast display leads to an specialised detail view (e.g. for an object the object view).
In the detail views you can again double-click on attributes, components in order to navigate to the subcomponents

By the way, the object view provides the possibility to display:

- The full inheritance relationship (including interfaces)
- Events and registered event handler
- Where used list (who points to this instance)
- ...

Variable Display II

Data Explorer


Data Objects Individ. Display

Name: BASE_TREE_CONTROL

Data Object	Type	Value
BASE_TREE_CONTR	Reference	{0:110*%CLASS=CL_GUI_COLUMN_TRE...
ACTIVE	C(1)	X
CATT_ACTIV	C(1)	
GUI_IS_RUNNIN	C(1)	X
H_GUI	Flat Structure(92)	Structure: flat & not charlike
IS_INIT	C(1)	X
JAVABEAN	C(1)	
WWW_ACTIVE	C(1)	
ADUST_DESIGN	I(4)	0
CUR_EVENT	Reference	{0:251*%CLASS=CL_GUI_EVENT}
ADUST_DESIGN_	I(4)	1
ALIGN_AT_BOTT	I(4)	8
PROPERTY_TAB	I(4)	240
ALIGN_AT_LEFT	I(4)	1
ALIGN_AT_RIGHT	I(4)	2
ALIGN_AT_TOP	I(4)	4
LIFETIME_DEFAU	I(4)	0
LIFETIME_DYNPF	I(4)	1
LIFETIME_IMODE	I(4)	2
METRIC_DEFAUL	I(4)	0
METRIC_MM	I(4)	2
METRIC_PIXEL	I(4)	1
MODE_DESIGN	I(4)	1
MODE_RUN	I(4)	0
PROPERTY_METI	I(4)	410
STATE_ALIVE	I(4)	0
STATE_ALIVE_ON	I(4)	1
STATE_DEAD	I(4)	1-
VISIBLE_FALSE	C(1)	0

© SAP AG 2005, 27

THE BEST-RUN BUSINESSES RUN SAP



- For nested objects, structures, references the data explorer allows you to dive into the depth without losing the context of the upper levels.

Loaded Programs - Globals

Display all global variables of the loaded programs

The screenshot shows the SAP Global Data interface. On the left, a navigation tree lists various loaded programs, with 'RSDBRUNT' selected. The main area displays a table of global variables for this program.

V	St	Variable Name	Val.	Technical Type	Hexadecimal Value
		DYN_SEL	Structure: deep	Deep Structure(24)	00000000FFFFFFFF00000000
		SCREEN_PROGS	Structure: deep	Deep Structure(240)	52534142415050524F4752
		SCREENS	Standard Table[0x23(216)]	Standard Table[0x23(216)]	
		SCR_STACK	Standard Table[0x26(640)]	Standard Table[0x26(640)]	
		CURRENT_SCREEN	Structure: deep	Deep Structure(216)	202020202020202020202020
		CURRENT_SCR	Structure: deep	Deep Structure(640)	202020202020202020202020
		PARENT_SCR	Structure: deep	Deep Structure(640)	202020202020202020202020
		PARENT_NO_SPAGPA	Standard Table[0x1(8)]	Standard Table[0x1(8)]	
		NO_SPAGPA_STACK	Standard Table[0x1(8)]	Standard Table[0x1(8)]	
		ANCESTORS_SCR	Standard Table[0x26(640)]	Standard Table[0x26(640)]	
		RESTORE	Structure: deep	Deep Structure(168)	000000002020202020202020
		MEMONUM	00000000	N(8)	3030303030303030
		VARISCREENS	Structure: flat & char11k	Flat Structure(5)	2020202020
		VSCR_TFILL	0	I(4)	00000000
		VSCR_INDEX	0	I(4)	00000000
		VSCREENS	Structure: flat & char11k	Flat Structure(4)	20202020
		VSCR_T	Standard Table[0x5(80)]	Standard Table[0x5(80)]	
		CURR_VSCR	Structure: deep	Deep Structure(80)	202020202020202020202020
		DYNS	Structure: flat & not char	Flat Structure(64)	202020202020202020202020
		DYNS_NODES	Structure: flat & char11k	Flat Structure(64)	202020202020202020202020
		DYNS_FIELDS	Structure: flat & not char	Flat Structure(76)	202020202020202020202020
		GL_VARIDYN	Standard Table[0x22(300)]	Standard Table[0x22(300)]	
		GL_VDATDYN	Standard Table[0x6(72)]	Standard Table[0x6(72)]	
		NOINTS	Structure: deep	Deep Structure(48)	202020202020202020202020

© SAP AG 2005, 29

THE BEST-RUN BUSINESSES RUN SAP™

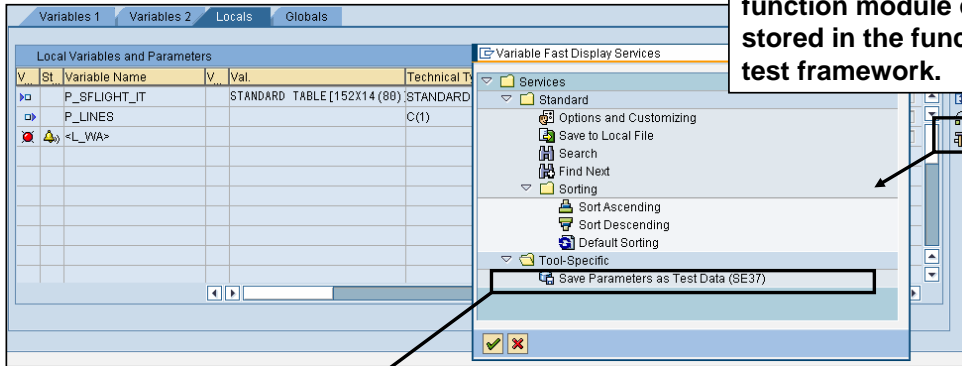


Two sections:

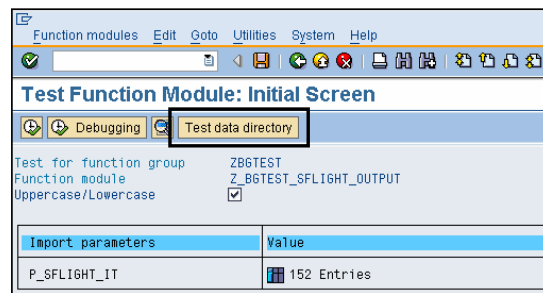
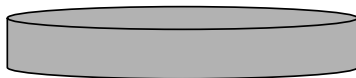
Loaded Programs: Here you see the attributes of all programs and the program groups
 Global Data : Get the global variables of each global variable just by double-clicking in the navigation tree.

Download Parameters To Function Builder

The current parameters of a function module can be stored in the function builder test framework.



Function builder test data dictionary



Variable Navigation: Source = Editor

```

15
16 loop at p_sflight_it assigning <l_wa> .
17   write: / <l_wa>-carrid, <l_wa>-connid, <l_wa>-fldate.
18   endloop.

```

Double click on variable

Navigation	
Navigation to Sec. Tools (e.g. Editor):	In a Parallel Window
Navigation to Detail Views:	Switch to Detail Tab
Variable Navigat. from Editor	Display Variable in Variable Fast Display
	Display Variable in Variable Fast Display
	Display Variable Directly in the Relevant Detail View

The screenshot shows the SAP ABAP Editor on the left with the code from the previous block. Below the editor is the 'Variables' window showing the variable P_SFLIGHT_IT with its value 'Standard Table[152x14(80)]'. On the right is the 'Detail Display' window showing a table of flight data.

Line	MANDT(0)	CARRID(0)	CONNID(4)	FLDATE(8)	PRICE(8) DE	CURRENCY(5)	PLANETYPE(1)	SEATSMAX(4)
1	000	AA	0017	20050311	1186.69	USD	747-400	660
2	000	AA	0017	20050312	1186.69	USD	747-400	660
3	000	AA	0017	20050313	1186.69	USD	747-400	660
4	000	AA	0017	20050314	1186.69	USD	747-400	660
5	000	AA	0017	20050315	1186.69	USD	747-400	660
6	000	AA	0017	20050316	1186.69	USD	747-400	660
7	000	AA	0017	20050317	1186.69	USD	747-400	660
8	000	AA	0017	20050318	1186.69	USD	747-400	660
9	000	AA	0064	20050311	1186.69	USD	A310-300	280
10	000	AA	0064	20050312	1186.69	USD	A310-300	280
11	000	AA	0064	20050313	1186.69	USD	A310-300	280
12	000	AA	0064	20050314	1186.69	USD	A310-300	280
13	000	AA	0064	20050315	1186.69	USD	A310-300	280
14	000	AA	0064	20050316	1186.69	USD	A310-300	280

© SAP AG 2005, 31

THE BEST-RUN BUSINESSES RUN SAP®



- If you use the new ABAP Editor then the quick info provides already the value of a variable.
Therefore if you want to get more details then in most cases (and especially if you run the editor in full screen mode) you want to navigate to the detail view instead of the variable fast display which will not deliver much more info.
In this case you should use the option: 'Variable Navigat. from Editor: Display Variable Directly in the Relevant Detail View.'
- As all options, you can save your settings so that with the next debugger start these settings are always active.

Variable Navigation: Source = Variable fast display

The screenshot illustrates the navigation options available when double-clicking on a variable in the variable fast display. A yellow callout box points to the variable `P_SFLIGHT_IT` in the variable fast display, with the instruction "Double click on variable".

The navigation menu offers the following options:

- Navigation to Sec. Tools (e.g. Editor): In a Parallel Window
- Navigation to Detail Views: Switch to Detail Tab
- Variable Navigat. from Editor: Create Detail View on Active Desktop
- Switch to Detail Tab
- Switch to Detail Tab if Tools Are Displaced

The screenshot shows the result of selecting "Switch to Detail Tab": the variable fast display is replaced by a table view of the variable's data. The table has columns: Line, MANDT(C), CARRID(C), CONNID(N), FLDATE(D), PRICE(P)DE, and CURRENCY(C). The data is as follows:

Line	MANDT(C)	CARRID(C)	CONNID(N)	FLDATE(D)	PRICE(P)DE	CURRENCY(C)
1	000	AA	0017	20050311	1186.69	USD
2	000	AA	0017	20050312	1186.69	USD
3	000	AA	0017	20050312	1186.69	USD
4	000	AA	0017	20050314	1186.69	USD
5	000	AA	0017	20050315	1186.69	USD
6	000	AA	0017	20050315	1186.69	USD
7	000	AA	0017	20050317	1186.69	USD
8	000	AA	0017	20050318	1186.69	USD
9	000	AA	0064	20050311	1186.69	USD
10	000	AA	0064	20050312	1186.69	USD
11	000	AA	0064	20050313	1186.69	USD
12	000	AA	0064	20050314	1186.69	USD
13	000	AA	0064	20050315	1186.69	USD
14	000	AA	0064	20050316	1186.69	USD

The screenshot also shows the source code of the variable, which is a loop that iterates over the variable `p_flight_it` and writes the current record to the output.

```

1  ** REFERENCE P_SFLIGHT_IT TYPE FLIGHTTAB
2  ** EXPORTING
3  ** REFERENCE P_LINES TYPE C
4  .....
11
12  field-symbol: <i_wa> type #flight.
13  .....
14
15
16  loop at p_flight_it assigning <i_wa> .
17    write / <i_wa>-carrid, <i_wa>-connid, <i_wa>-fldate.
18  endloop.
19

```

© SAP AG 2005, 32

THE BEST-RUN BUSINESSES RUN SAP®



- If you double click on a variable in the variable fast display then per default you navigate to the tab which hosts the appropriate detail view (e.g. table view)
- If you prefer to create the detail view on the current desktop please choose: Navigation to Detail Views: Create Detail View on Active Desktop.



- Motivation
- Architecture & Handling
- UI Basics
- New ABAP Frontend Editor
- Variable Display & Navigation
- **Comparing Variables (Diff-Tool)**
- Breakpoints & Watchpoints

Diff Tool – Navigation To Connected Detail Views

The screenshot shows the SAP ABAP Debugger interface. At the top, the menu bar includes Debugger, Edit, Goto, Breakpoints, Settings, Miscellaneous, System, and Help. The main window title is "(1) - ABAP Debugger Controls Session 1 (Exclusive)". Below the title bar, there are several tabs: Desktop 1, Desktop 2, Desktop 3, Standard, Structures, Tables, Objects, Detail Displs., Break/Watchpoints, and Diff. The "Diff" tab is active, showing a comparison between two variables: Variable 1 (ITAB) and Variable 2 (ITAB_COPY). Both are of type "Standard Table[152x14(80)]". The diff tool has identified three differences, all related to "Different Table Keys". The first difference is in the "PRICE" field, where the value is 1234.00 in ITAB and 1186.69 in ITAB_COPY. Below the diff list, two detail views are shown side-by-side. The left view is for table ITAB, and the right view is for table ITAB_COPY. Both views show a table with columns: Line, PRICE/P(8) DE, CURRENCY[C(5)], and PLANETYPE[C(1)]. The price difference is clearly visible in the first row of each table.

Index	Diff	Description	Location	ITAB
1	!	Different Table Keys	ITAB[1]-MANDT	Key
2	!	Different Table Keys	ITAB[1]-CURRENCY	Key
3	!	Different Table Keys	ITAB[1]-PLANETYPE	Key
4		The elements have different contents:	ITAB[5]-PRICE	1234.00

Table	Table Type	Standard Table[152x14(80)]
ITAB	Standard Table[152x14(80)]	
Line	PRICE/P(8) DE	CURRENCY[C(5)] PLANETYPE[C(1)]
5	1234.00	USD 747-400
6	1186.69	USD 747-400
7	1186.69	USD 747-400
8	1186.69	USD 747-400
9	1186.69	USD A310-300
10	1186.69	USD A310-300
11	1186.69	USD A310-300

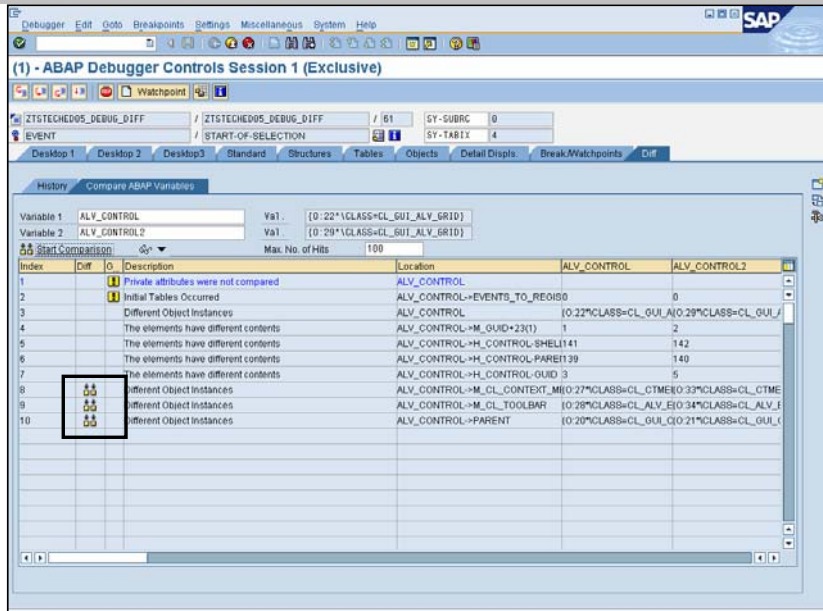
Table	Table Type	Standard Table[152x14(80)]
ITAB_COPY	Standard Table[152x14(80)]	
Line	PRICE/P(8) DE	CURRENCY[C(5)] PLANETYPE[C(1)]
5	1186.69	USD 747-400
6	1186.69	USD 747-400
7	1186.69	USD 747-400
8	1186.69	USD 747-400
9	1186.69	USD A310-300
10	1186.69	USD A310-300
11	1186.69	USD A310-300

© SAP AG 2005, 35


THE BEST-RUN BUSINESSES RUN SAP®

- ◆ Just by double-clicking on a hit in the diff result list the variables are displayed in parallel and the debugger navigates to this difference.

Diff Tool – Start New Diff For Subcomponents



For subcomponents of type reference or internal table no “deep diff” is executed.

But in the corresponding result lines you find again a diff button  to start a comparison for these sub components.

You may use the history tab to get back to the first result list.

© SAP AG 2005, 36

THE BEST-RUN BUSINESSES RUN SAP®



- ◆ In order to assure a good performance of the diff tool even for deeply nested variables we execute no deep diff.
- ◆ This means if you have e.g. an internal table in a structure then we only check if the tables of both structures are equal but you will not get details where they differ. To get these details you can run a diff for these subcomponents.



- Motivation
- Architecture & Handling
- UI Basics
- New ABAP Frontend Editor
- Variable Display & Navigation
- Comparing Variables (Diff-Tool)
- **Breakpoints & Watchpoints**

How many different breakpoint kinds do exist ?



Debugger breakpoints

- scope: debugging session
- set in : debugger

Perhaps 4. type:
`BREAK <User-Name> .`
`BREAK-POINT .`



Session breakpoints

- scope: logon session
- set in : development workbench

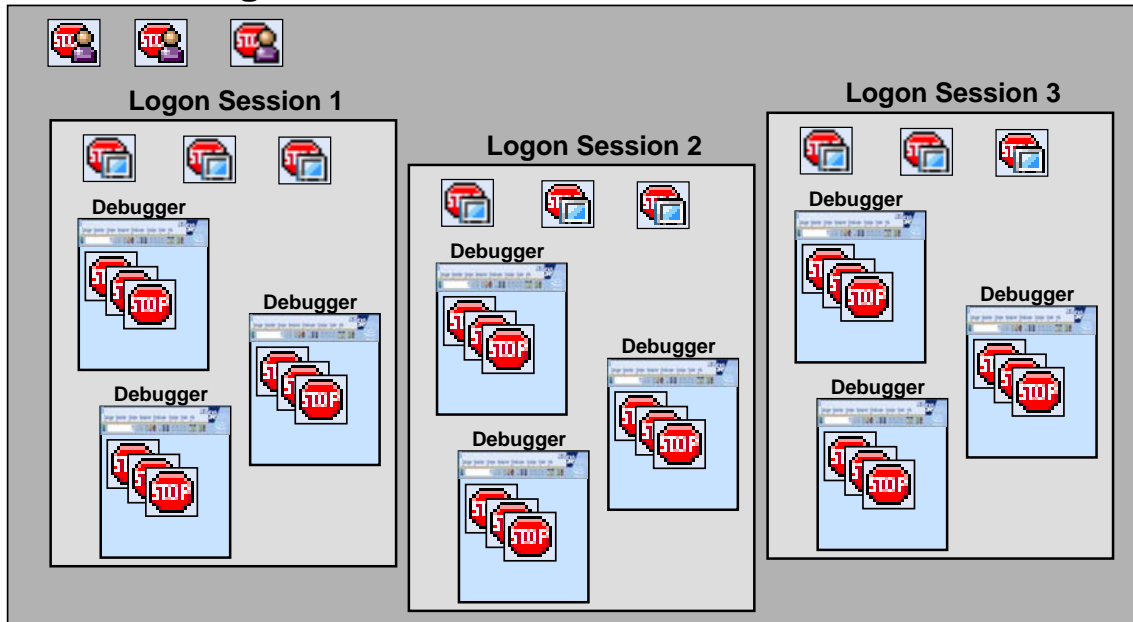


User (external) breakpoints

- scope: all logon sessions (for one user on one server)
- set in : development workbench

- In the classic debugger Debugger breakpoints are only available in the current internal session. By saving the breakpoints they were converted to Session breakpoints.
- In the New ABAP Debugger the Debugger breakpoints are available for all internal sessions of the current external session.
- In the classic debugger User(External) breakpoints can only be used for RFC and http sessions. In the New ABAP Debugger you can use User(External) for all kind of process types.

All logon sessions of one user on one server



Breakpoints – Create A Breakpoint

Create/Delete a breakpoint in the Editor by clicking on the breakpoint column

The screenshot displays the SAP ABAP Debugger interface. The main editor shows the source code for the method `handle_expand_no_children`. A 'Create Breakpoints' dialog is open, allowing the user to select a class and method. The class `CL_TPDA_SERVICES` is selected, and a list of methods is shown, including `DYNP_MESSAGE`, `ERROR_LOG`, `HANDLE_CX_ROOT`, and `INIT_MESS_LOG_FLAG`. A callout box points to the 'Create Breakpoints' dialog with the text: 'F4 help on classes / methods / function modules / ...'. Another callout box points to the 'Method name' field in the dialog with the text: 'F4 help on classes / methods / function modules / ...'. The status bar at the bottom indicates the scope is `IMETHOD handle_expand_no_children`.

Breakpoints – Maintain Breakpoints

Create/Delete/Activate/Deactivate breakpoints

Poi.	Actv.	N	Visibility	Breakpnt Type	Breakpoint at	Skip	Include
				Session Breakpoint	Source Code	INCL=ZTSTECHE05_DEBUG_EX_1 (55) / PRG=Z	ZTSTECHE05_DEBUG_E
				Session Breakpoint	Source Code	INCL=ZTSTECHE05_DEBUG_EX_2 (31) / PRG=Z	ZTSTECHE05_DEBUG_E
				Session Breakpoint	Source Code	INCL=CL_TREE_MODEL=====CM00H	CL_TREE_MODEL=====
				Session Breakpoint	Source Code	INCL=CL_TREE_MODEL=====CM00H	CL_TREE_MODEL=====
				External (User) Br	Source Code	INCL=CL_TREE_MODEL=====CM00H	CL_TREE_MODEL=====
				Debugger Breakpoint	ABAP Command	SELECT	

Specify how often the breakpoint shall be skipped before stopping

© SAP AG 2005, 41

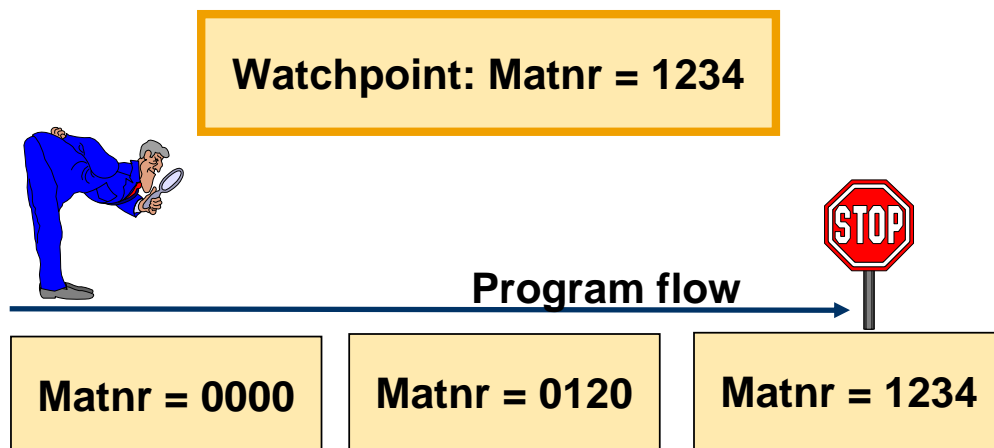
THE BEST-RUN BUSINESSES RUN SAP® 

- In the break-point maintenance view you can easily delete/create/activate/deactivate your breakpoints.
- Additionally you may convert your breakpoints from debugger to session breakpoints etc.
- Double-clicking on the navigation button displays the breakpoint in the source code.

Watchpoints

If you set a watchpoint on the field `matnr` (`matnr = '1234'`), the debugger stops as soon as `matnr` contains the value '1234'.

If you do not specify a condition, the debugger stops as soon as the field `matnr` changes.



© SAP AG 2005, 42

THE BEST-RUN BUSINESSES RUN SAP™

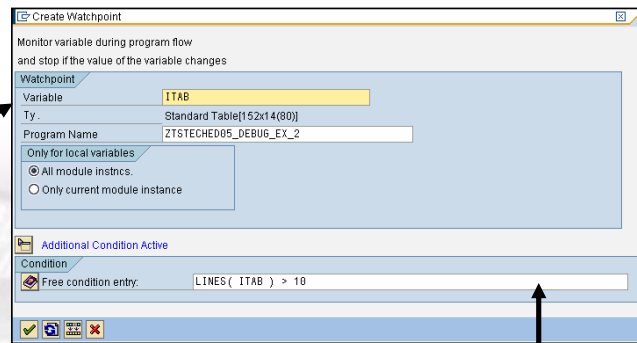


- Watchpoints are very dynamic breakpoints. You can specify a field that should be monitored and you can specify a stop condition.
- The watchpoints of the New ABAP Debugger are not shared with the Classic Debugger. This is a totally new implementation which assures correct behavior of the watchpoints for all kind of variables (including object attributes, local variables and even internal tables)

Create a Watchpoint

- **specify variable**

- stop on change of variable
- all global, local variables
- including internal tables
- not possible: debugger symbols pointing into internal tables
e.g.: "itab[1]-comp"



- **specify variable & condition**

- stop when variable changes AND condition is true
- condition restricted to two operands and 1 operator
- built-in functions ("lines()" and "strlen()") possible
e.g.: "sy-subrc <> 0" or "strlen(string) > 10"

- can be activated / inactivated

© SAP AG 2005, 43

THE BEST-RUN BUSINESSES RUN SAP™



Classic Debugger: No watchpoints on internal tables allowed, but...

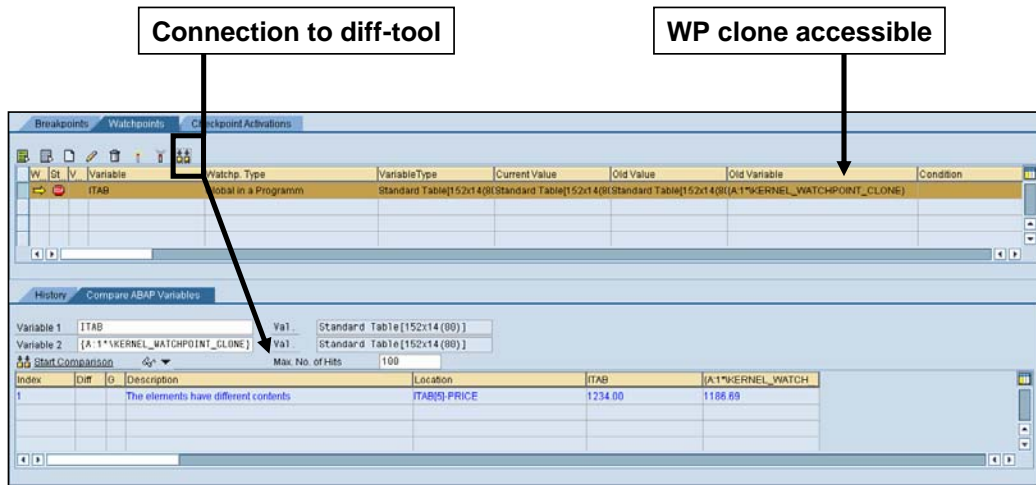
- You can use a watchpoint on the table header of an internal table to watch at least the line count of an internal table:
- Each internal table has a header which contains administration data like number of lines.
If, e.g., someone deletes lines or appends lines, then this header is updated. It is possible to set a watchpoint on the table header. This is very helpful if you want, e.g., find the source line where an internal-table line is deleted. But keep in mind that modifications of the contents in a line have normally no effect on the table header.
There is some more info stored in the table header (like the loop counter); therefore don't be too astonished if the program stops, e.g., at a LOOP statement.
- Watchpoints on table headers in different kernel releases:

< 40B	itab-*sys*
40B	itab[]
45B	itab[]+0(128)
since 46D	*itab[] (header is only filled when table is not initial!)

Since kernel release 46D the table header is no longer stored directly in the addressable memory. Only a table reference⁴³ (8bytes) remains in the memory addressable by the user. This table reference is a pointer which points to the header

When stopping at a watchpoint

- value of “old” variable is available (reference to WP clone)
- Diff-tool can be used to be guided to difference(s)



© SAP AG 2005, 44

THE BEST-RUN BUSINESSES RUN SAP™



- In contrast to the Classic Debugger the New ABAP Debugger creates always a clone of the watch-point variable .
- After reaching a watchpoint the clone contains the status of the old variable and the current variable contains the current status.
- Using this clone technique, you can now compare these two variables –e.g. by using the diff tool to find out where the variable was changed.

- No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.
 - Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.
 - Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.
 - IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, and Informix are trademarks or registered trademarks of IBM Corporation in the United States and/or other countries.
 - Oracle is a registered trademark of Oracle Corporation.
 - UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.
 - Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.
 - HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.
 - Java is a registered trademark of Sun Microsystems, Inc.
 - JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.
 - MaxDB is a trademark of MySQL AB, Sweden.
 - SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.
-
- The information in this document is proprietary to SAP. No part of this document may be reproduced, copied, or transmitted in any form or for any purpose without the express prior written permission of SAP AG.
 - This document is a preliminary version and not subject to your license agreement or any other agreement with SAP. This document contains only intended strategies, developments, and functionalities of the SAP® product and is not intended to be binding upon SAP to any particular course of business, product strategy, and/or development. Please note that this document is subject to change and may be changed by SAP at any time without notice.
 - SAP assumes no responsibility for errors or omissions in this document. SAP does not warrant the accuracy or completeness of the information, text, graphics, links, or other items contained within this material. This document is provided without a warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.
 - SAP shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials. This limitation shall not apply in cases of intent or gross negligence.
 - The statutory liability for personal injury and defective products is not affected. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third-party Web pages nor provide any warranty whatsoever relating to third-party Web pages.