



| z/TPF V1.1

TPF Users Group – Fall 2012

z/TPF Debugger Update

Josh Wisniewski
Development Tools Subcommittee

AIM Enterprise Platform Software
IBM z/Transaction Processing Facility Enterprise Edition 1.1.0

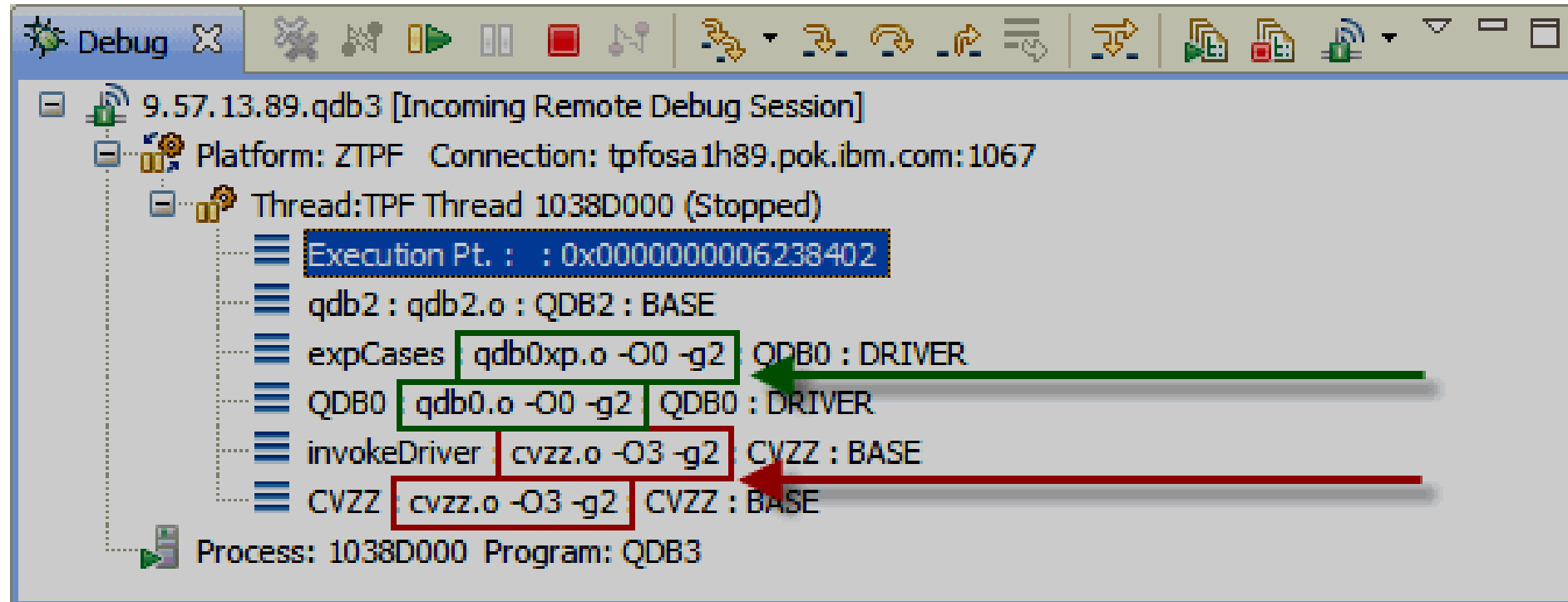
Any reference to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.

Agenda

- **New Features**
 - **Show Code Optimization Level**
 - **Show Loadset Name**
 - **Fork Interface Enhancement**
 - **Memory Search**
 - **Improve Remote Debug Information**
 - **Dump Viewer Data Scrubbing User Exit**
 - **CDB0 Interface**
 - **Prevent ZINET ADD S-DEBUG**
 - **Remote Debug Information Name User Exit**
- **Previously released features**
 - **Code Coverage Tool**
 - **Auto detect workstation IP**
 - **Improve debugger connection failure handling**

Show Code Optimization Level

- Debug view shows optimization level and debug information level (dwarf level) for each object in each stack frame.
 - Assembler objects do not show settings (for example qdb2.o).
 - C/C+ objects should be built with `-O0` for optimal debugging.



The screenshot shows the Debug View interface with the following stack frames and their optimization levels:

- Execution Pt. : : 0x00000000006238402
- qdb2 : qdb2.o : QDB2 : BASE
- expCases : qdb0xp.o -O0 -g2 QDB0 : DRIVER
- QDB0 : qdb0.o -O0 -g2 QDB0 : DRIVER
- invokeDriver : cvzz.o -O3 -g2 CVZZ : BASE
- CVZZ : cvzz.o -O3 -g2 CVZZ : BASE

Process: 1038D000 Program: QDB3

Green arrows point from the optimization level text to the object name in the frame above. Red arrows point from the optimization level text to the object name in the frame below.

Show Loadset Name

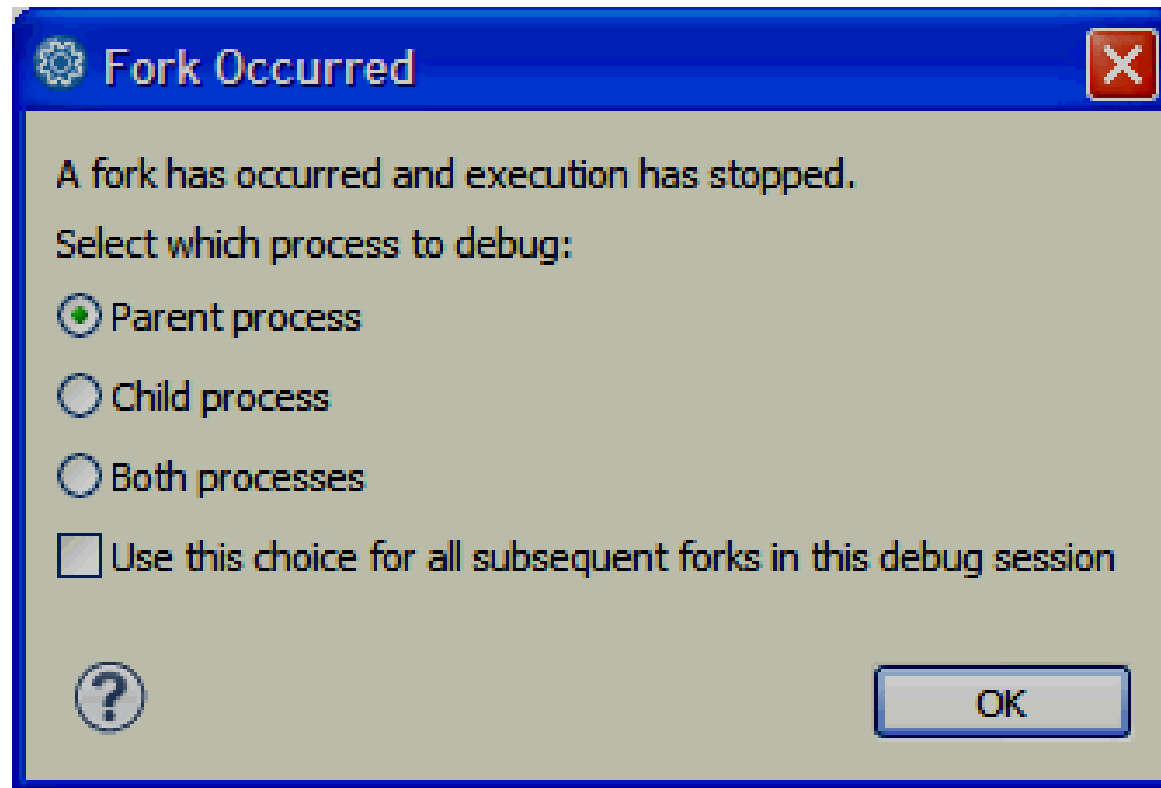
- Debug view shows the loadset name for each module in each stack frame.
 - Modules without a loadset name will show as BASE.

Debug View Screenshot:

- 9.57.13.89.qdb3 [Incoming Remote Debug Session]
- Platform: ZTPF Connection: tpfosa1h89.pok.ibm.com:1067
- Thread: TPF Thread 1038D000 (Stopped)
- Execution Pt. : 0x0000000006238402
- qdb2 : qdb2.o : QDB2 **BASE**
- expCases : qdb0xp.o -O0 -g2 : QDB0 **DRIVER** ←
- QDB0 : qdb0.o -O0 -g2 : QDB0 **DRIVER**
- invokeDriver : cvzz.o -O3 -g2 : CVZZ **BASE**
- CVZZ : cvzz.o -O3 -g2 : CVZZ **BASE**
- Process: 1038D000 Program: QDB3

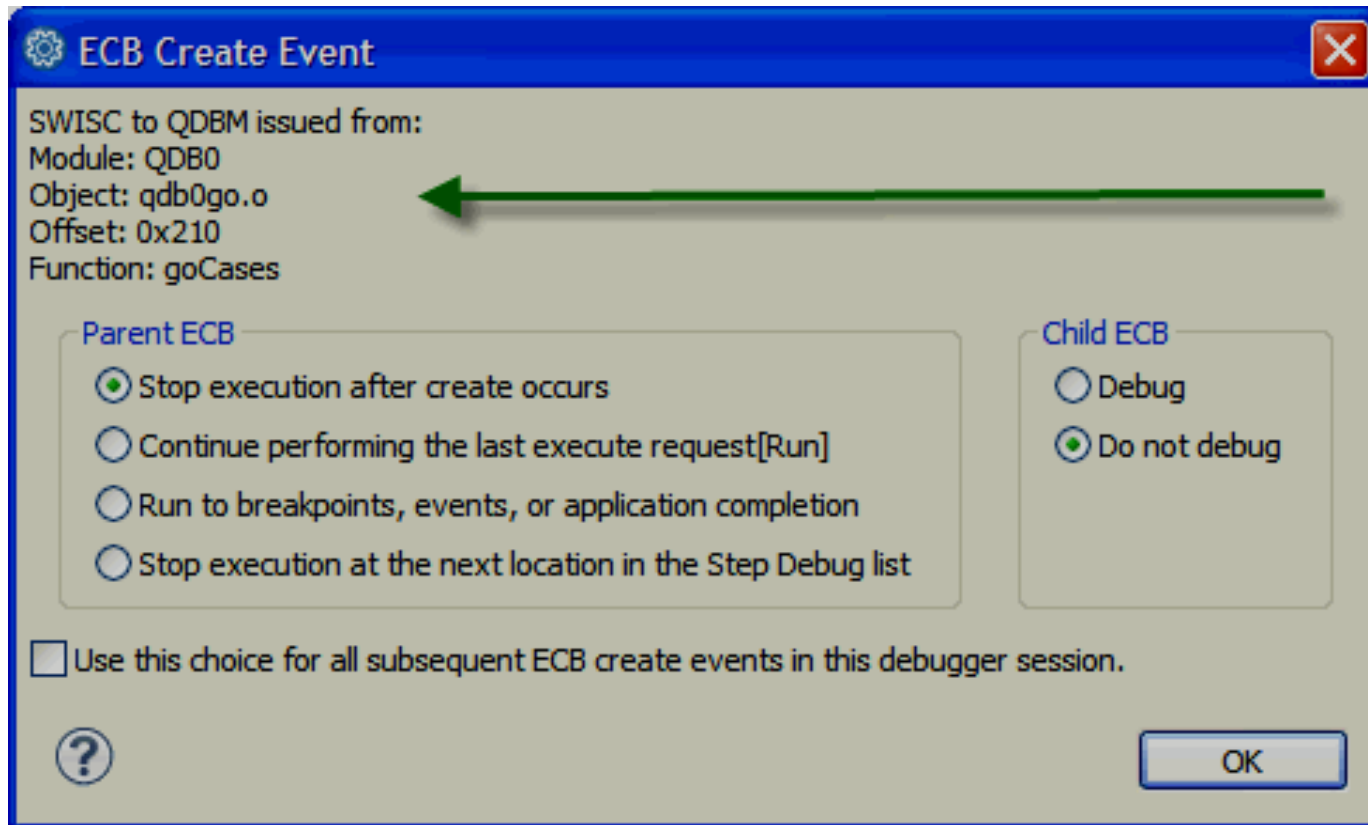
Fork Interface Enhancement

- The old fork interface presents options with ambiguous meaning.
- Details regarding the fork event are shown in the Debug Console which may be hidden.



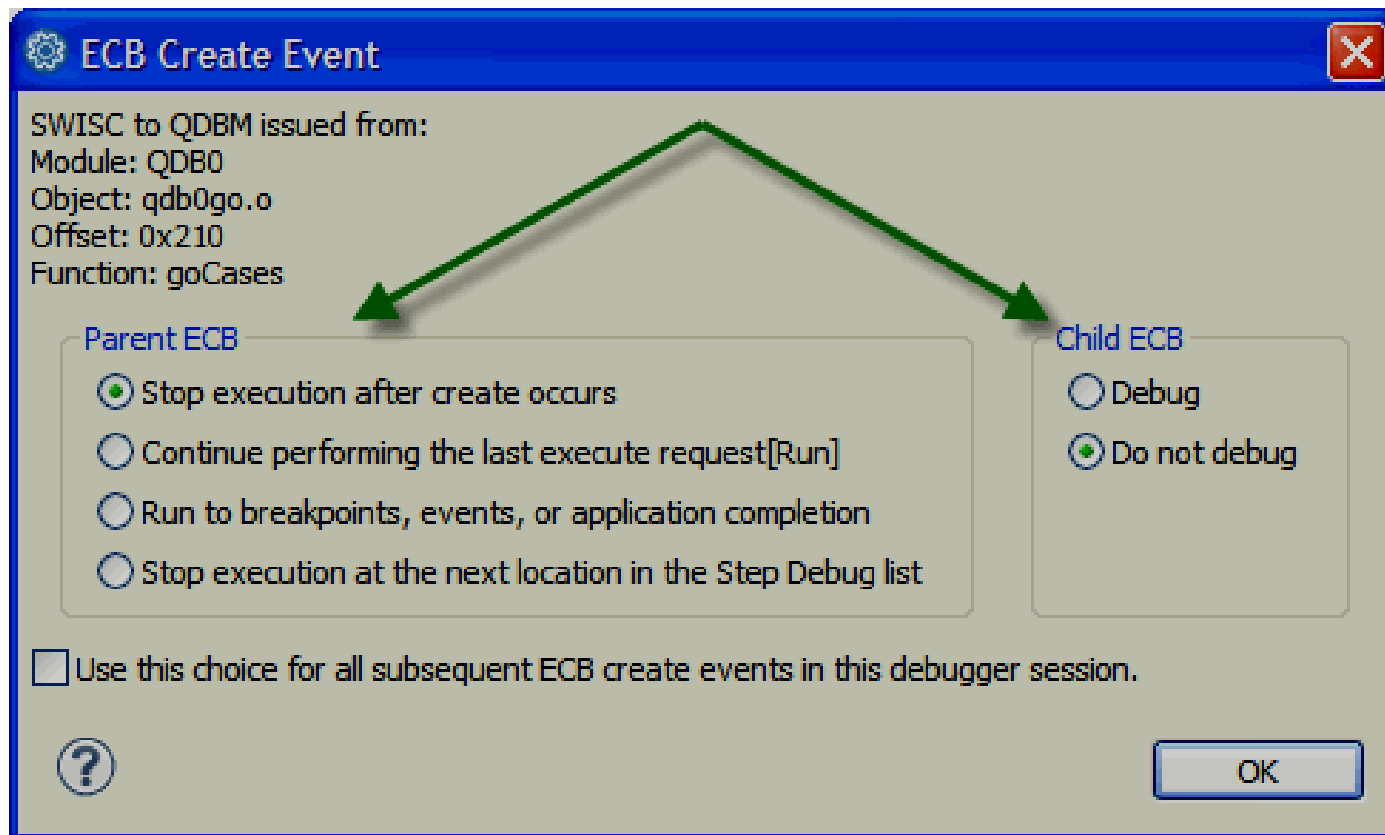
Fork Interface Enhancement

- The new ECB Create interface shows concise information.
 - What type of ECB Create event occurred.
 - Which module will be entered by the child ECB.
 - Where the ECB Create event occurred.



Fork Interface Enhancement

- The new ECB Create interface provides concise choices for both the parent and child ECBs separately.
- Two new options provided for parent ECB: Run and Step Debug.
- Hover over an option for more information.



Memory Search

- Right click and choose “Search Memory” in any memory view.
- Specify search pattern and type: HEX, ASCII, EBCDIC, UTF-8.
- Specify search range.

The screenshot shows the IBM z/OS Debug Console interface. The main window displays a memory dump for address 10300000. A 'Search' dialog box is open, allowing the user to specify search parameters. The dialog box contains the following fields and options:

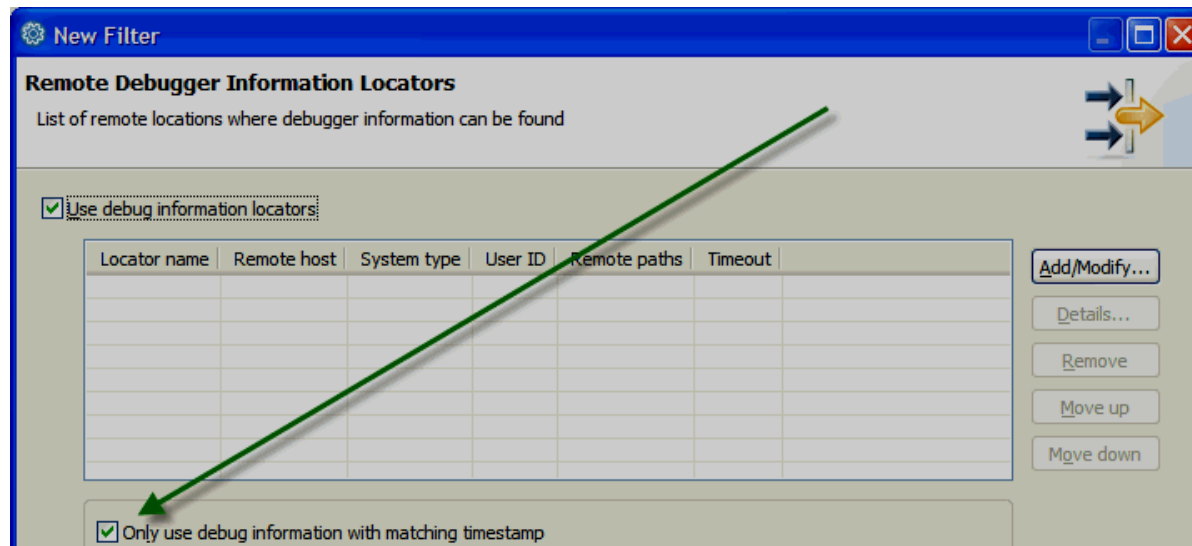
- Search pattern:** FFFFFFFF
- Pattern format:** Hex
- Starting address:** 0x10300000
- Memory length:** 0x1000
- Wrap search
- Buttons: Find Previous, Find Next, Close

The memory dump in the background shows the following data:

Address	0 - 3	4 - 7	8 - B	C - F
0000000010300210	00000000	00000000	00000000	FF000000
0000000010300220	00000000	FFFFFFFF	CA4B3849	688FA5C0
0000000010300230	00000000	00000000	00000000	053931E0
0000000010300240	00000000	00000000	00000000	00000000
0000000010300250	00000000	00000000	00000000	00000000
0000000010300260	00000000	00000000	00000000	00000000
0000000010300270	00000000	00000000	00000000	00000000
0000000010300280	00000000	00000000	10302E3C	00000000
0000000010300290	00000000	0240A000	02412000	00000000
00000000103002A0	00000000	00000000	00000000	00000000
00000000103002B0	00000000	00000000	00000000	00000000
00000000103002C0	00000000	00000000	052C9260	00000000
00000000103002D0	00000000	00000000	060EF4DE	00010000
00000000103002E0	00000000	00000000	00000000	00000000
00000000103002F0	00000000	00000000	00000000	00000000
0032000010300300	00000000	00000000	00000000	00000000
0000000010300310	00000000	10300288	00000000	00000000
0000000010300320	00000000	00000000	00FF1C00	00000000
0000000010300330	00000000	000000C2	FFFFFFFF	00000000
FF00000010300340	00000000	00000000	00000000	00000000
0000000010300350	00010000	39000000	00010000	00000000

Improve Remote Debug Information

- The Remote Debug Information feature allows you to store your debug information on a remote system as selected at registration time for the debugger to dynamically load to z/TPF as needed.
- This new feature allows the debugger to search multiple remote paths for debug information that exactly matches the code loaded to the system. If an exact match is not found, no debug information is used to help ensure users do not have a debug information mismatch.
- Turning off this feature allows you to override the debug information as you can today.



Dump Viewer Data Scrubbing User Exit

- The debugger dump capture process has been enhanced to call user exit `CDBX_ScrubDataUserExit` in `cdbxds.asm` for all data areas captured to allow you to scrub any data from the dumps.
- This may be useful if you are not using the z/TPF Non-Displayable ECB Storage feature (NDSPEC).

CDB0 Interface

- This new feature externalizes the interface to CDB0 as `tpf_flag_for_debug` which allows you to flag ECBs for debugging when an ECB is started from a custom Comms package.

Prevent ZINET ADD S-DEBUG

- This feature ensures that the z/TPF debugger daemon listener is defined on z/TPF as DBUG to ensure that the debugger daemon recycling (ZINET STOP and ZOLDR commands) occurs correctly.

Remote Debug Information Name User Exit

- This new user exit allows you to compose the name of shared objects to locate when using the Remote Debug Information feature.
- This may be useful if you do not use the maketpf build solution.

Code Coverage Tool

- Typical use cases of the code coverage tool:
 - A QA manager needs to know which modules of an application are not executed when a driver suite is run (hundreds or thousands of modules).
 - A tester needs to ensure that a test plan drives all modules, objects, and functions in an application (tens of modules).
 - A developer needs to ensure that all lines of a new application component have been tested (a few modules at most).
 - A QA regression test team is tasked with cooperatively writing a driver to test every line of an application.
- Diagnostic use cases of the code coverage tool:
 - Determine execution paths taken through ported code.

Code Coverage Tool

- Similar registration process to the z/TPF debugger.
- Program centric so multiple ECBs can cause data to be collected simultaneously in the same program.
- Size analysis provides instruction level statistics at the module, object and function level.
- Source analysis provides source line statistics at the source file and function level.

New Filter

New Code Coverage Registration Session

Enter information for code coverage registration

Workstation Information

Workstation name * Workstation TCP/IP address *

TPF Terminal

Terminal name *

LNIATA IP Address LU Name

Program Mask

QD*

Note: Code coverage details will only be collected for the programs specified in the list above.

User token

Code Coverage Settings

Subsystem Name (optional):

Root Save Directory Name (optional):

Example: Specify 'tmp' to save code coverage data in '/tmp'.

Automatically Perform Size Analysis

Automatically Perform Source Analysis

Session registration

Automatically register new session upon creation

Code Coverage Tool

- Code coverage view shows execution statistics with sorting, filtering, export and etc capabilities.
- Editor view shows which lines have and have not been executed.
- Demo viewlet available at www.ibm.com/support/docview.wss?uid=swg21570467

The screenshot displays the IBM Code Coverage Tool interface. On the left, a tree view shows the project structure with files like `qdb0.cpp` and `qdb0bp.cpp`. A table provides execution statistics for these files.

File	Size Percentage	Line Percentage
Host: 9.57.13.89, Session: qdb, Timestamp: October 9, 2012 9:4		
QDB0	8%	7%
iuddi.hpp [1 function covered. Results may be incomplete.]	can not be determined	not executed
qdb0.cpp	14%	18%
callJava	not executed	not executed
dispHelp	not executed	not executed
QDB0	16%	19%
viewDump	not executed	not executed
qdb0bp.cpp	not executed	not executed
qdb0er.cpp	not executed	not executed
qdb0ff.cpp	not executed	not executed
qdb0go.cpp	41%	51%
qdb0lk.cpp	not executed	not executed
qdb0ms.cpp	not executed	not executed
qdb0sk.cpp	not executed	not executed
qdb0sy.cpp	not executed	not executed
qdb0th.cpp	not executed	not executed
qdb0xp.cpp	30%	29%
string.h [2 functions covered. Results may be incomplete.]	can not be determined	2%
tpfregs.h [3 functions covered. Results may be incomplete.]	can not be determined	not executed
QDB2	23%	87%
QDB3	3%	100%
QDBA	not executed	not executed
QDBB	not executed	not executed

On the right, the code editor shows the source code for `qdb0.cpp`. Lines 118, 120, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, and 142 are highlighted in green, indicating they were executed. Lines 119, 121, 122, 123, 124, 125, 126, 133, 134, 135, 136, 137, 138, and 140 are highlighted in red, indicating they were not executed.

```

116
117 /* display help manual if parser error */
118 if (num_parms < 1)
119 {
120     dispHelp();
121 }
122
123 /******
124 /*
125 /*
126 /******
127
128 for ( i = 1, cur_parm = &parse_results; i <= num_parms;
129     i++ , cur_parm = cur_parm->IPRSE_next )
130 {
131     if(strcmp(cur_parm->IPRSE_parameter,"ERR-d++")==0)
132     {
133         type = QDBGDRV_ERR;
134         testcase=atoi(cur_parm->IPRSE_value);
135         ecbptr()->ebw000 = testcase;
136         ecbptr()->ebw001 = type;
137         continue;
138     } /* end of error */
139     else if(strcmp(cur_parm->IPRSE_parameter,"GO-d++")==0)
140     {
141         type = QDBGDRV_GO;
142         testcase=atoi(cur_parm->IPRSE_value);

```

Code Coverage Tool

- New Comparison feature to quickly see the differences between runs.
- Comparison feature includes an editor comparison tool to see the source analysis differences in a source file.

The screenshot displays the Code Coverage Tool interface, divided into two main panes.

Left Pane: Compare code coverage results

Table 1: File Comparison

File	Size 1	Size 2
qdb0th.cpp	n/a	n/a
qdb0xp.cpp	6	+40
array_test()	n/a	+81
class_test()	n/a	+78
DataLevelFill_test()	n/a	n/a
expCases	26	+1
FillAndHold_test()	n/a	n/a
scalar_test()	n/a	n/a
struct_test()	n/a	+84
string.h	n/a	n/a
tpfregs.h	n/a	n/a
ODR2	23	0

Table 2: Line Comparison

File	Line 1	Line 2
qdb0th.cpp	n/a	n/a
qdb0xp.cpp	5	+41
array_test()	n/a	+100
class_test()	n/a	+100
DataLevelFill_test()	n/a	n/a
expCases	21	+10
FillAndHold_test()	n/a	n/a
scalar_test()	n/a	n/a
struct_test()	n/a	+100
string.h	2	0
tpfregs.h	n/a	n/a
ODR2	87	0

Right Pane: Code Coverage Source Analysis Compare

This pane shows a side-by-side comparison of source code. The left column shows the original code, and the right column shows the code after analysis. The code is color-coded to highlight differences: green for code that is present in both, red for code that is present in the original but not in the analysis, and grey for code that is present in the analysis but not in the original.

```

440
441 QDB0_printf("executing debugger EX
442
443 switch (caseNum)
444 {
445     case 1:
446         QDB2 (&reg);
447         break;
448
449     case 2:
450         array_test ();
451         break;
452
453     case 3:
454         struct_test ();
455         break;
456
457     case 4:
458         class_test ();
459         break;
460
461     case 5:
462         scalar_test ();
463         break;
464
465     case 6:
466         QDBD1F3 (type, caseNum);
  
```

Auto detect workstation IP

- The debugger attempts to automatically detect the workstation IP address and correct situations where the user has specified the workstation IP address incorrectly.
- Demo viewlet available at www.ibm.com/support/docview.wss?uid=swg21570467

Improve debugger connection failure handling

- On connection failures, the debugger attempts to notify the TPF Toolkit that registered the debugger of the debugger connection failure.
- ADB01 dumps indicate a connection failure occurred. Some ADB01 dumps did not include any information that was helpful in debugging the cause of the dump. As such, these ADB01 dumps were converted to WTOPC messages with additional information to aid in the diagnosis of the problem.

z/TPF Debugger Deliverable Details

Description	z/TPF APAR	z/TPF PUT Level	TPF Toolkit Level	TPFUG Requirement
CDB0 Interface	PJ39617	PUT9	None	Customer Request
Remote Debug Info Name			None	Customer Request
User Exit				
Dump Viewer Data Scrubbing			None	Customer Request
User Exit				
Show Loadset Name			None	Customer Request
Show Code Optimization Level			None	Customer Request
Prevent ZINET ADD S-DEBUG			None	Customer Request
Improve Remote Debug Info			V3.6.4	Customer Request
Memory Search			V3.6.4	V09114F

z/TPF Debugger Deliverable Details

Description	z/TPF APAR	z/TPF PUT Level	TPF Toolkit Level	TPFUG Requirement
Fork Interface Enhancement	PJ40255	PUT9	V.next	Customer Request
Code Coverage Tool: Size Analysis Source Analysis Comparison Feature	PJ37973 PJ38995	PUT8	V3.6 V3.6.3 V.next	
Auto detect workstation IP Improve debugger connection failure handling	PJ38995	PUT8	V3.6.3	Customer Request

Trademarks

- IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “[Copyright and trademark information](http://www.ibm.com/legal/copytrade.shtml)” at www.ibm.com/legal/copytrade.shtml.

Notes

- Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.
- All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.
- This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.
- All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.
- Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.
- Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.
- This presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.