*TPF Users Group*
*Grapevine, Texas*

IBM Software Group

# TPF Debugger

Enhancements for z/TPF

*Application Development Subcommittee*

Daniel Gritter / IBM TPF Development

October 2004

**AIM Core and Enterprise Solutions**

IBM z/Transaction Processing Facility Enterprise Edition 1.1.0

## New Features

- TPF Toolkit platform
- Dump Viewing under the Debugger
- ECB Snapshot viewing
- Enhanced Optimized code debugging
- Preprocessor Macro Support
- New Console commands
- New Disassembler
- Debugging in 1052 state

## TPF Toolkit platform

- The IBM TPF Toolkit for WebSphere studio will be the only supported interface for the debugger under z/TPF 1.1
- Features an fully configurable/customizable integrated debugging/development environment, allowing seamless integration from design to verification.
- Interactive learning environment using step-by-step tutorials to guide users on how to perform common tasks
- Extensive help selection on configuration and development procedures

## Dump Viewing under the Debugger

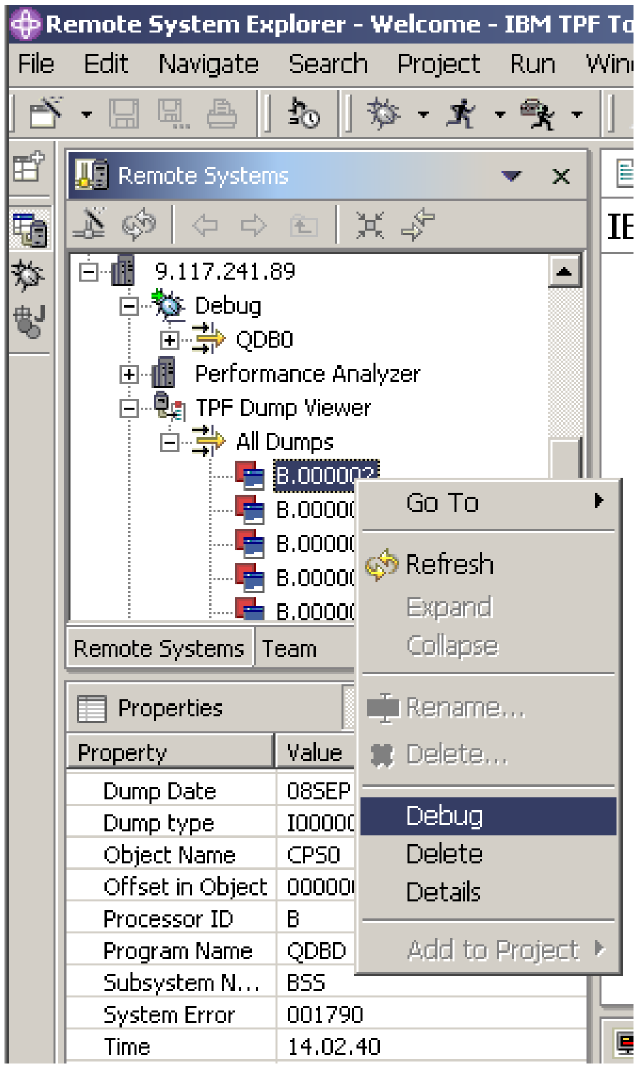
- Allows you to view an application dump that occurs on your TPF system in an interactive debugging environment
  - Display ECB stack
  - Display modules, parts, functions in source view
  - Display C/C++ and Assembler Variables/Expressions
  - Display contents of storage at the time of dump
  - Display registers at the time of dump
- Quick convenient method of diagnosing application problems without the hassle of searching through pages of listings and dump screens
- Integrated into the IBM TPF Toolkit for WebSphere Studio
- Integrated management utilities allow for maintenance free operation, including a policing routine to clean up expired dump files after a specified interval
- Import option allows you to take a dump on one TPF system and view it from another TPF system

## How to use Dump Viewing

- ZASER DBUG must be specified to turn on this feature, ZASER NODBUG may be used to turn it off
- NODUMP/NODUPL options also apply to dump viewing files
- Any application controlled SERRC or unplanned dump will cause a dump file to be created in the file system on TPF.
- Capturing dumps in 1052 state requires pools to be active in 1052 state
  - ZPOOL 1052 UP

## How to use Dump Viewing

- TPF Toolkit is used to bring up a dump for viewing
  - A summary list is presented of dumps captured, along with a properties display for the highlighted dump
  - You can create filters to select which dumps are displayed, based on selectiong criteria, including program name, subsystem, dump type, and processor ID.
  - Three options, Debug, Delete, Details
    - **Debug** brings up the dump in the debugger view
    - **Delete** removes the dump from TPF
    - **Details** shows the register values/PSW at time of dump



| Property | Value |
|---|---|
| Dump Date | 08SEP |
| Dump type | I00000 |
| Object Name | CPS0 |
| Offset in Object | 00000 |
| Processor ID | B |
| Program Name | QDBD |
| Subsystem N... | BSS |
| System Error | 001790 |
| Time | 14.02.40 |

## Managing Dump Files

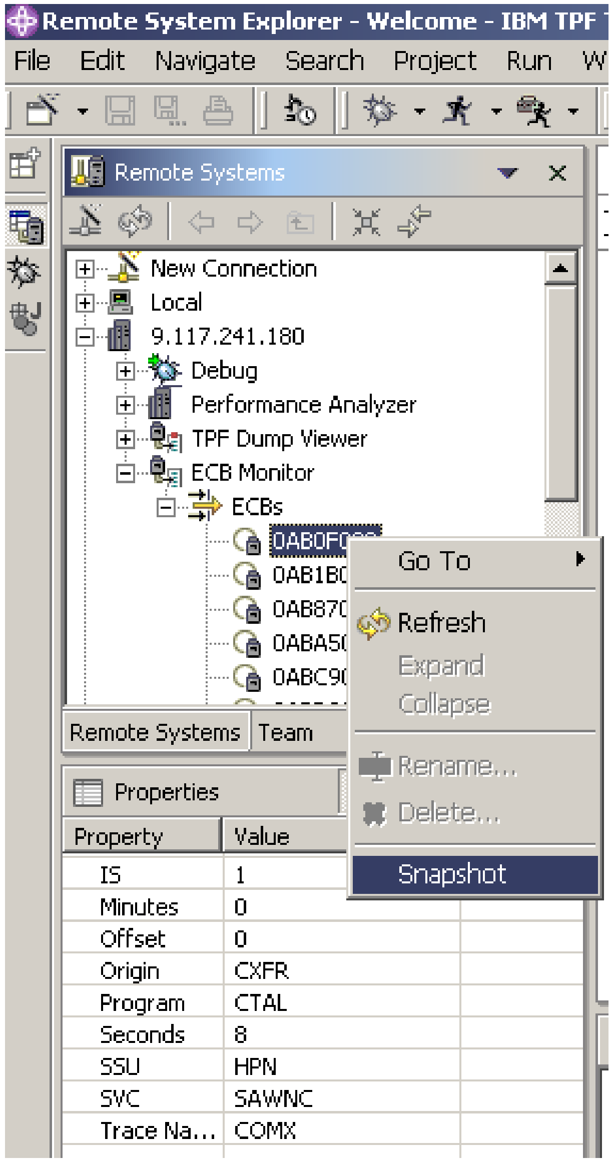
- A policing ECB is created every 30 minutes to scan for expired dumps
- Files stored under the /dbgdumpX directory on the filesystem, where X is the processor ID.
- ZDDMP functional message can be used for manage dump files
  - ZDDMP IMPORT
    - Allows you to import a dump file that was captured on a different TPF system for viewing
  - ZDDMP ALTER
    - Specifies the dump retention period (default 24 hours)
  - ZDDMP SYNC
    - Synchronizes the dump control records with the dump files present on the file system
    - Allows management of dump files using ZFILE commands
  - ZDDMP CLEAN
    - Forces the policing routine to run and removes any expired dump files

## Viewing Dump Files

- ZDDMP DISPLAY
  - Allows you to view details of a stored dump from TPF console
  - Filters available for:
    - Complex
    - Date
    - Dump Number
    - Program
    - Processor
    - System Error Number
    - Subsystem ID
    - Terminal
- ZDDMP DELETE
    - Allows you to delete dumps based on the same filter criteria

## ECB Monitor

- ECB Monitor is a tool used to diagnose behavior of long running ECBs
- Integrated into the IBM TPF Toolkit for WebSphere Studio
- Opens a debug session where you can view the variables, ECB trace, stack, and source view of the application execution point
- Allows you to save a snapshot for side-by-side comparison in a different debug session, using the **ECBSnapshot** command from the Debug Console view.
- Summary list of ECBs is displayed, with the properties pane displaying the details for the selected ECB.
- Snapshot action is available to monitor the selected ECB.



| Property | Value |
|---|---|
| IS | 1 |
| Minutes | 0 |
| Offset | 0 |
| Origin | CXFR |
| Program | CTAL |
| Seconds | 8 |
| SSU | HPN |
| SVC | SAWNC |
| Trace Na... | COMX |

## Enhanced Optimized Code Debugging

- Previously (PUT17 support), optimized code could be debugged in source view, however variables and expressions were not available for viewing
- Added support for viewing variables in optimized code
- Some variables may not be available if they have been optimized out by the compiler
- For initial logic debugging unoptimized code is still optimal, as it avoids instruction reordering which causes the execution to appear to "jump around" in source view

## Preprocessor Macro Support

- The gcc compiler allows the user to include the preprocessor macros in the debug info
- Adding the -g3 option to your compile options will include this information
- When the -g3 option is used the developer can use C Preprocessor macros in expression evaluation
  - For example, if you had the following macro in your code:
    - #define MAX(x,y)  x > y ? x : y
  - You could then enter an expression using the macro
    - MAX(1,2)
  - Which would then display as the result the value 2
- This is primarily useful when using a structure such as the ecbptr()  that has lots of defines, allowing you to use these defines to reference the field names you used while coding.
- This option will typically increase the size of the debugging information significantly

## New Debug Console commands (ECB TRACE)

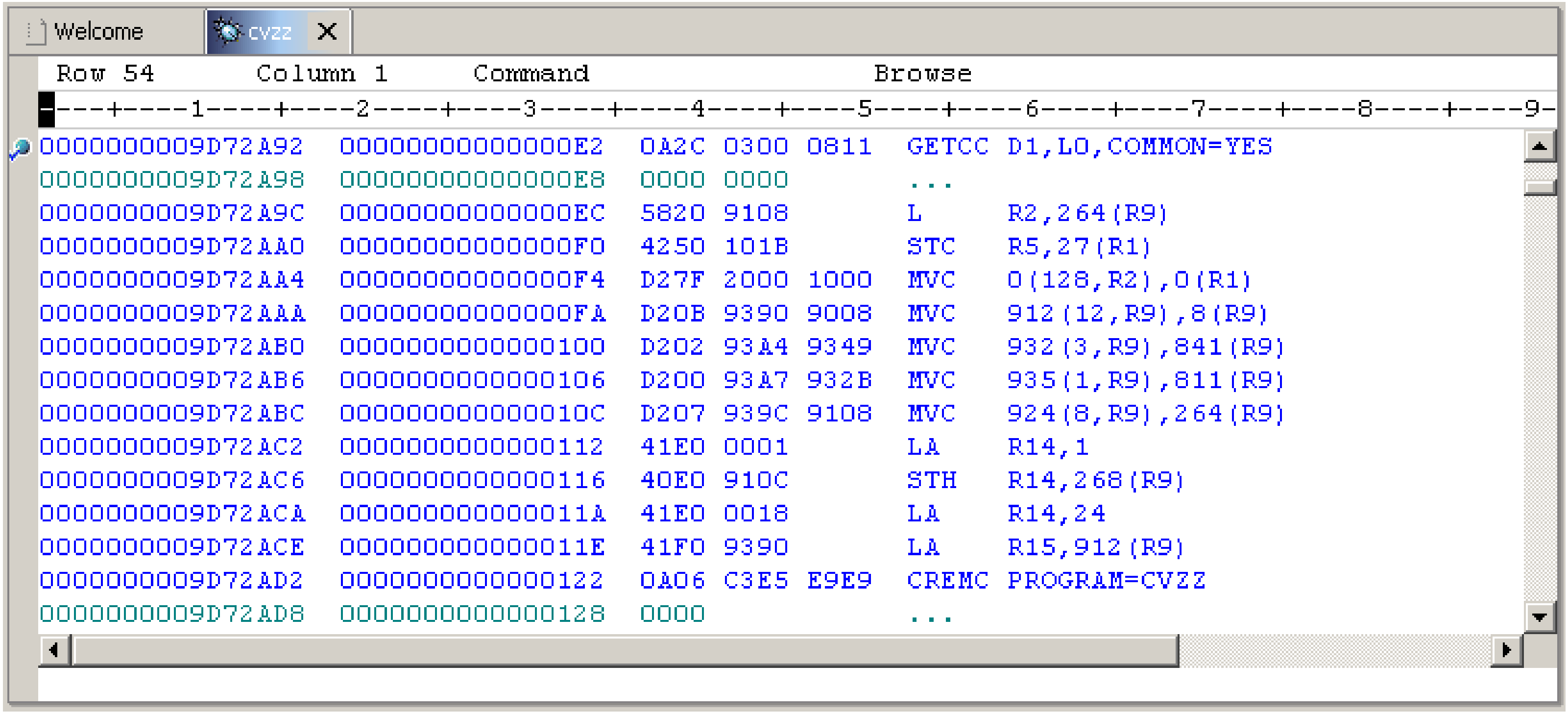
- ECB Trace displays a summary of the most recent functions that have been executed by the ECB, along with parameters and return codes
- ECB Trace display commands
  - ECBTRACE VERBOSE
  - ECBTRACE TRGROUP
- Provides ZDECB TR support from the debugger view without the need for setting up any special LNIATA support.
- Used to display ECB trace entries for the ECB being debugged
- Shows Collated Macro/C function trace
- Available in Dump Viewing, ECB Monitor, and traditional debugging modes

## New Debug Console commands (TRACE LOG)

- The trace log facility writes a copy of the ECB trace information to tape or to a file.
- The new  TRLOG command allows you to activate/deactivate the trace log from a debugger session
  - This allows you to limit the scope of trace log using debugger breakpoints, allowing you to refine the amount of data that is collected.
- TRLOG DIR-xxx
  - activate trace log for the APPL ECB, send the trace to file
- TRLOG TAPE-xxx
  - activate trace log for the APPL ECB, send the output to tape
- TRLOG OFF
  - turn off trace log for the APPL ECB

## Updated Disassembler

- Support complete set of z/Architecture instructions
- Added macro parameter decoding for TPF SVC macros
- ZDCOR/ZDPGM have also been updated to use the new disassembler
- Example below of GETCC/CREMC parameters displayed:



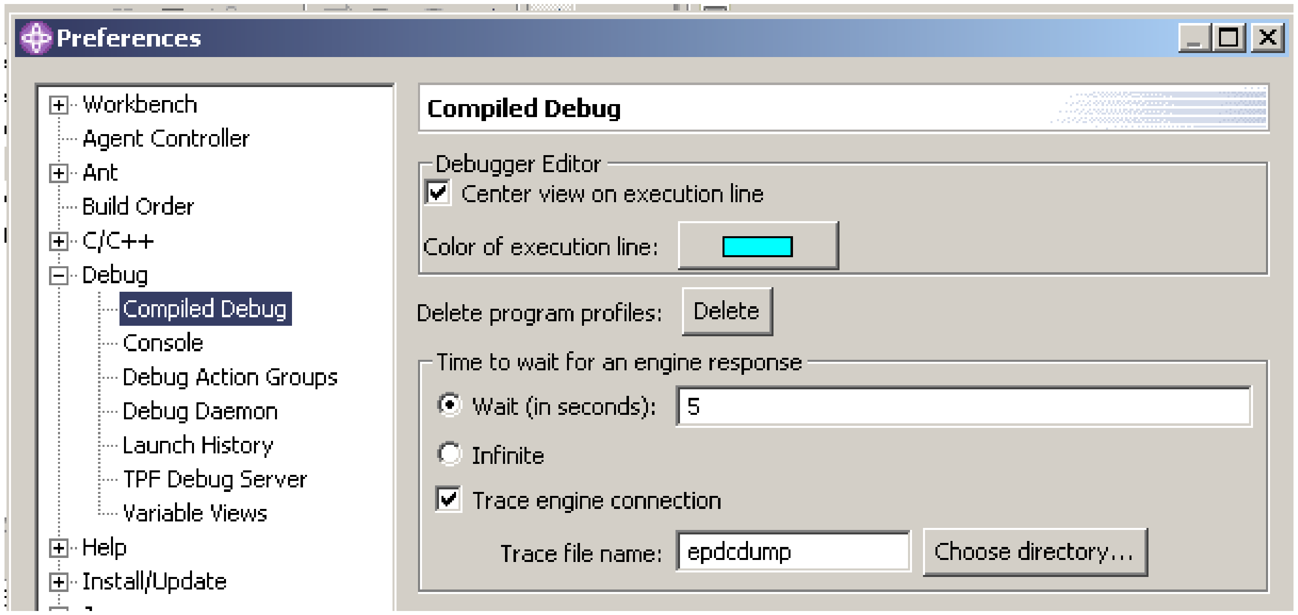
```
 Row 54        Column 1        Command                      Browse
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9-
000000009D72A92   00000000000000E2   0A2C 0300 0811   GETCC  D1,L0,COMMON=YES
000000009D72A98   00000000000000E8   0000 0000        ...
000000009D72A9C   00000000000000EC   5820 9108        L      R2,264(R9)
000000009D72AA0   00000000000000F0   4250 101B        STC    R5,27(R1)
000000009D72AA4   00000000000000F4   D27F 2000 1000   MVC    0(128,R2),0(R1)
000000009D72AAA   00000000000000FA   D20B 9390 9008   MVC    912(12,R9),8(R9)
000000009D72AB0   0000000000000100   D202 93A4 9349   MVC    932(3,R9),841(R9)
000000009D72AB6   0000000000000106   D200 93A7 932B   MVC    935(1,R9),811(R9)
000000009D72ABC   000000000000010C   D207 939C 9108   MVC    924(8,R9),264(R9)
000000009D72AC2   0000000000000112   41E0 0001        LA     R14,1
000000009D72AC6   0000000000000116   40E0 910C        STH    R14,268(R9)
000000009D72ACA   000000000000011A   41E0 0018        LA     R14,24
000000009D72ACE   000000000000011E   41F0 9390        LA     R15,912(R9)
000000009D72AD2   0000000000000122   0A06 C3E5 E9E9   CREMC  PROGRAM=CVZZ
000000009D72AD8   0000000000000128   0000             ...
```

## Debugging in 1052 state

- You can now debug applications running in 1052 state
- Debug server must first be activated in 1052 state
  - ZINET alt s-dbug state-1052
- OSA Polling must be active in 1052 state
  - zosae modify osa-xxxx state-1052
- From this point on, you can register your program and debug as normal
- Does not require 1052 state pools for debugger

## Enhanced Serviceability Features

- New Debug Console command to turn on internal debugger tracing
- TPFDBgtrace [Value-]
  - turn on/off TPF debugger trace info
  - trace value is defined in iudtr.h
- Option to turn on EPDC trace part of IBM TPF Toolkit for WebSphere Studio
  - Found under Compiled Debug preferences: Trace Engine Connection



Compiled Debug

Debugger Editor
[✓] Center view on execution line
Color of execution line: 🟦

Delete program profiles: Delete

Time to wait for an engine response
● Wait (in seconds): 5
○ Infinite
[✓] Trace engine connection
Trace file name: epdcdump

## Recap

- Dump Viewing under the Debugger
- ECB Snapshot viewing
- Enhanced Optimized code debugging
- Preprocessor Macro Support
- New Console commands
- New Disassembler
- Debugging in 1052 state

- Any questions?

## Legal