WebSphere. software

IBM Debug Tool for z/OS, V11.1

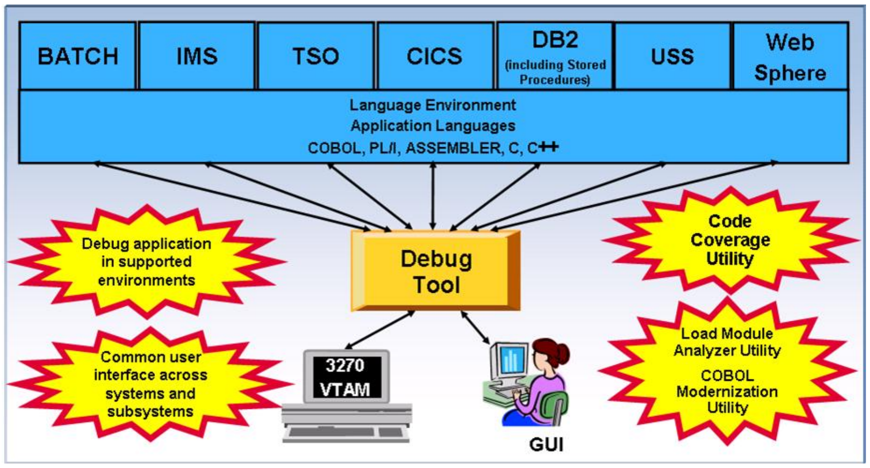

BATCH
IMS
TSO
CICS
DB2 (including Stored Procedures)
USS
Web Sphere
Language Environment
Application Languages
COBOL, PL/I, ASSEMBLER, C, C++
Debug application in supported environments
Debug Tool
Code Coverage Utility
Common user interface across systems and subsystems
3270 VTAM
GUI
Load Module Analyzer Utility
COBOL Modernization Utility

Figure 1: Debug Tool environment

Highlights

- ***Provides a single debugging tool for batch, TSO, CICS, DB2,DB2 stored procedures and IMS applications written in COBOL,PL/I, C/C++ and assembler***
- ***Offers more productivity enhancements when used with Rational Developer for System z (available separately)***
- ***Includes tools to quickly identify and convert OS/VS COBOL code to ANSI 85 standard***
- ***Supplies tools to help you determine how thoroughly your code has been tested***
- ***GUI plug-in available for download***

In an increasingly complex and competitive environment with challenging business demands, application developers are faced with constant pressure to deliver function-rich applications quickly.

Regardless of whether it was designed to perform routine or critical tasks, the underlying code that drives your applications is highly complex. Application developers have to work quickly to meet demand, with minimal errors — even adapting code on the fly as your business needs evolve.

To effectively build and service applications, you need robust, easy-to-use tools to compile, test and debug your applications.

IBM® Debug Tool for z/OS®, V11.1 is an interactive source-level debugging tool for compiled applications. It is a program testing and analysis aid that helps you examine, monitor, and control the execution of applications written in C, C++, COBOL, PL/I or assembler on a z/OS system.

It provides debugging capability for applications running in a variety of environments, such as batch, TSO, IBM CICS®, IBM IMS™, IBM DB2®, IBM DB2 stored procedures and IBM z/OS UNIX® System Services.

Debug Tool also includes features to help you identify old OS/VS and VS COBOL II applications and to upgrade the source code automatically to IBM Enterprise COBOL. It also supplies tools to help you determine how thoroughly your code has been tested.

Debug Tool for z/OS, V11.1 replaces all prior versions of both IBM Debug Tool for z/OS and IBM Debug Tool Utilities and Advanced Functions for z/OS. This single Debug Tool for z/OS, V11.1 product includes all of the function in the previous separate products as well as the new V11.1 function. Delivering this single comprehensive product provides significantly more function to existing Debug Tool for z/OS customers, and will help simplify ordering and installation.

Debug Tool is tightly integrated with IBM Rational® Developer for System z® and other tools in the IBM portfolio of problem determination tools, so that you can develop, test and debug traditional and SOA applications from the same user interface.

Data at your fingertips

Debug Tool provides an interactive, full-screen, IBM 3270 system-based terminal interface with four windows that enable single-step debugging, dynamic patching and breakpoints:

- *The Monitor window displays the status of items you select, variables and registers. You can view, monitor and alter application variables or storage in real-time.*
- *The Source window displays the program code, highlighting the statement being run. In the prefix area of this window, you can enter commands to set, display and remove breakpoints.*
- *The Log window records and displays your interactions with Debug Tool and can show program output. The information you see in this window is included in the log file.*
- *The Memory window (swappable with the Log window) helps you display and scroll through sections of memory. You can update memory by typing over existing data with new data. The Memory window also keeps track of addresses for easier navigation.*

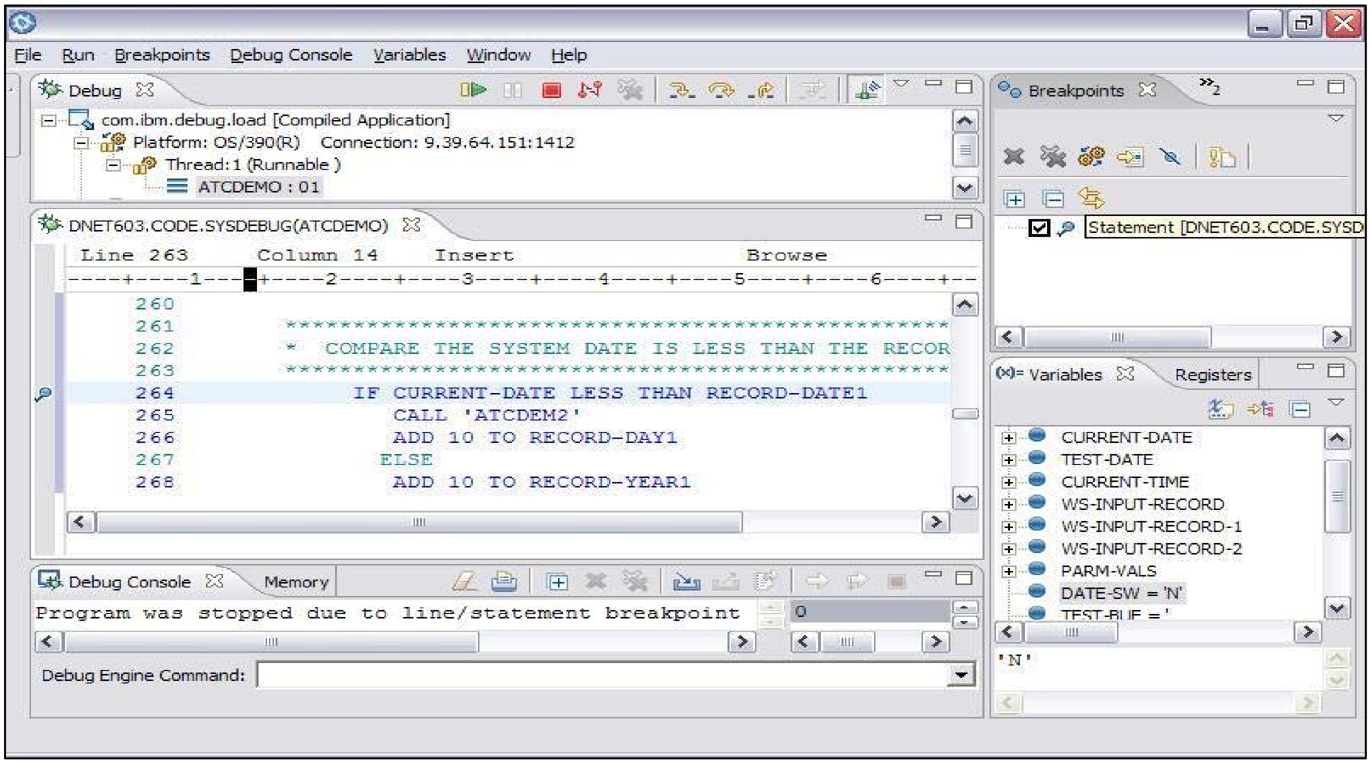
Debug Tool gives you great flexibility to choose how to display monitored variables and lets you update large or small variables directly in the monitor window.

For COBOL character variables displayed using the automonitor command, Debug Tool displays values in character format regardless of whether the string contains unprintable characters. You can change these values by typing over them in the Monitor window.
In the automonitor section of the Monitor window, you can display the value of a variable in the variable's declared data type and displaying the user register names in assembler AUTOMONITOR output, when possible.

Control debugging environment

With Debug Tool, you can choose how you view and manage the process of debugging your applications. Using the full-screen interface, you can interactively debug almost any application as it runs — including batch applications.

You can start Debug Tool when an application starts or during an abend. Alternatively, you can write applications so that they start the tool automatically — at specified times — interrupting the running of the application.



```
com.ibm.debug.load [Compiled Application]
  Platform: OS/390(R)   Connection: 9.39.64.151:1412
    Thread:1 (Runnable )
      ATCDEMO : 01
```

DNET603.CODE.SYSDEBUG(ATCDEMO)

```
 Line 263      Column 14     Insert                   Browse
----+----1----+----2----+----3----+----4----+----5----+----6----+--

        **************************************************
        *  COMPARE THE SYSTEM DATE IS LESS THAN THE RECOR
        **************************************************
             IF CURRENT-DATE LESS THAN RECORD-DATE1
                CALL 'ATCDEM2'
                ADD 10 TO RECORD-DAY1
              ELSE
                ADD 10 TO RECORD-YEAR1
```

Statement [DNET603.CODE.SYSD

```
Program was stopped due to line/statement breakpoint
```

```
CURRENT-DATE
TEST-DATE
CURRENT-TIME
WS-INPUT-RECORD
WS-INPUT-RECORD-1
WS-INPUT-RECORD-2
PARM-VALS
DATE-SW = 'N'
TEST-BUF = '
'N'
```

Figure 2: Rational Developer for System z works with Debug Tool to help mainframe developers be more productive.

Using the setup utility, you can create a setup file that contains the program information you need — including file allocations, runtime options, program parameters and application name — to run your application and start Debug Tool.

Setup files can save you time when you are debugging a program that you have to restart multiple times. You can create several setup files for each program. Each setup file can store information about starting and running your program under different circumstances.

IBM Language Environment® user exits can be linked with the application or with a private copy of a Common Execution Environment (CEE) runtime load module.

Review source while you debug

Debug Tool enables you to focus on a particular problem area by checking your application for errors one line at a time. By using single-step debugging — and setting dynamic breakpoints — you can monitor, interrupt and continue the flow of the application to identify errors easily.

A basic breakpoint indicates a stopping point in your program. For example, you can use a breakpoint to stop on a particular line of code. Breakpoints can also contain instructions, calculations and application changes. For example, you can set a breakpoint to have Debug Tool display the contents of a variable when the debugging process reaches a particular line of code.

You can also use a breakpoint to patch the flow of the program dynamically. You can set breakpoints in an application to monitor variables for changes, and watch for specified exceptions and conditions while an application runs. You can set, change and remove breakpoints as you debug the application. This means that you don't have to know where you want to set a breakpoint before you start debugging.

In CICS, Debug Tool supports "pattern matching breakpoints" that use the program or compile unit names specified in CADP or DTCN profiles to start Debug Tool and provides commands to enable and disable the breakpoints.

You can also debug applications written in a mix of COBOL, C, C++ or PL/I languages without leaving the tool. You can also include assembler programs in this mix and, using the disassembly view, you can debug programs compiled with the NOTEST compiler option or applications that include other languages.

For each programming language you can use a set of interpreted commands to specify actions to be taken. These commands are subsets of the languages — so they're easy to learn, and you can modify the flow of your application while you are debugging it. You can use the commands to dynamically patch (or alter) the value of variables and structures and to control the flow of an application.

SOA development and debugging

Debug Tool supports debugging of monolithic, composite, and SOA applications. Customers creating new Web services — whether newly written or refactored using existing application assets that use Rational Developer for System z — can immediately debug them using the Debug Tool plug-in provided.

DESCRIBE CHANNEL and LIST CONTAINER commands can display CICS channels and containers, including containers that hold state information for Web services. Users can display the information, even if it is not being referenced by the application program being debugged.

Debug Tool now provides support for invoking the z/OS XML parser to parse complete XML 1.0 or 1.1 documents in memory. If the document is syntactically valid, the XML is formatted and shown in the Debug Tool log. Otherwise, diagnostic information is provided to help identify the syntax error.

Enhanced debugging capabilities

Debug Tool provides a rich set of commands, tools and utilities to help you to debug your programs. When used with the setup utility in Debug Tool, these can help to:
Prepare your high-level language and programs for debugging by converting, compiling (or assembling) and linking your COBOL, PL/I, C/C++ and assembler source code.
-Conduct analysis on your test cases to determine how thoroughly they validate your programs.

In complex applications, it's easy to forget how you reached a particular point in your program. Debug Tool commands enable you to replay statements that have already run. If you compile your program with the IBM COBOL for OS/390® and VM compiler, or the Enterprise COBOL for z/OS compiler, you can review the values of variables and replay the statements while debugging.

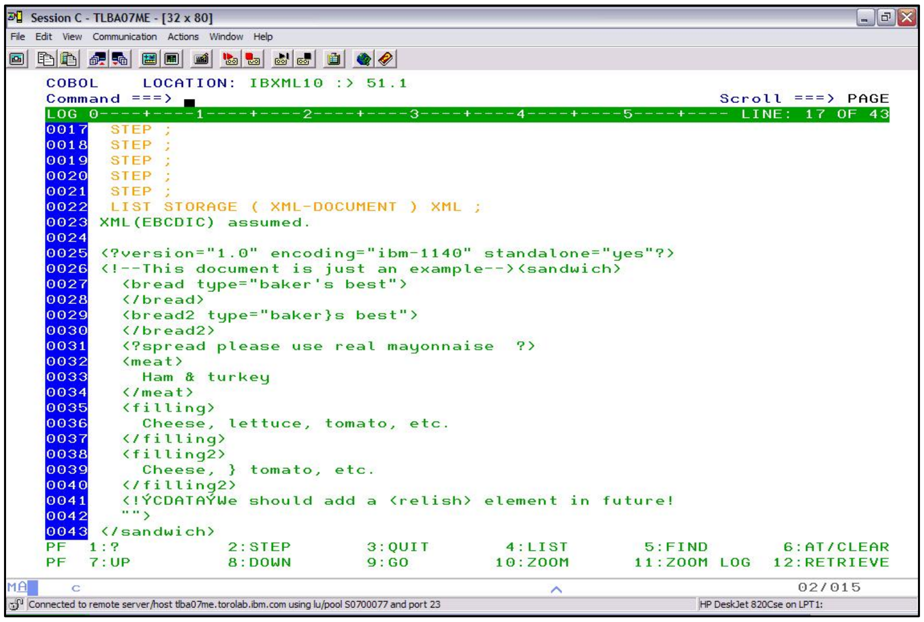
For programs compiled with the COBOL for OS/390 and VM compiler, the Enterprise COBOL for z/OS compiler, or the Enterprise PL/I for z/OS compiler, you can automatically monitor the values of variables referenced at the current statement. When the automonitor function is active, any variables that are referenced by the current statement are automatically selected for monitoring. You can view these variables in the monitor window.

Move to Enterprise COBOL to reuse and extend existing code

Previously, to create faster, more efficient applications, you had to sacrifice debugging support. With Debug Tool you can debug Enterprise COBOL for z/OS applications that have been compiled with standard or full-optimization compiler options.

You can also analyze your load modules to help you identify candidate OS/VS COBOL programs for conversion and then to convert these OS/VS COBOL applications to Enterprise COBOL for z/OS. You can then compile and debug these applications to extend the life of your existing code.

Debug Tool software also provides coverage tools that enable you to conduct analysis on your test cases and determine how thoroughly they exercise your programs.


Session C - TLBA07ME - [32 x 80]
File Edit View Communication Actions Window Help

```
COBOL     LOCATION: IBXML10 :> 51.1
Command ===>                                              Scroll ===> PAGE
LOG 0----+----1----+----2----+----3----+----4----+----5----+---- LINE: 17 OF 43
0017  STEP ;
0018  STEP ;
0019  STEP ;
0020  STEP ;
0021  STEP ;
0022  LIST STORAGE ( XML-DOCUMENT ) XML ;
0023 XML(EBCDIC) assumed.
0024
0025 <?version="1.0" encoding="ibm-1140" standalone="yes"?>
0026 <!--This document is just an example--><sandwich>
0027   <bread type="baker's best">
0028   </bread>
0029   <bread2 type="baker}s best">
0030   </bread2>
0031   <?spread please use real mayonnaise  ?>
0032   <meat>
0033     Ham & turkey
0034   </meat>
0035   <filling>
0036     Cheese, lettuce, tomato, etc.
0037   </filling>
0038   <filling2>
0039     Cheese, } tomato, etc.
0040   </filling2>
0041   <!ÝCDATAÝWe should add a <relish> element in future!
0042   "">
0043 </sandwich>
PF  1:?          2:STEP      3:QUIT      4:LIST      5:FIND      6:AT/CLEAR
PF  7:UP         8:DOWN      9:GO       10:ZOOM     11:ZOOM LOG 12:RETRIEVE
MA     c                                                          02/015
Connected to remote server/host tlba07me.torolab.ibm.com using lu/pool S0700077 and port 23    HP DeskJet 820Cse on LPT1:
```

Figure 3: Formatted XML structure using List Storage command

Combine with other development tools to optimize applications

Debug Tool shares a number of side files with IBM Fault Analyzer, making it easier for you to test and manage abends in new and existing applications. For example, the IDILANGX file produced by Fault Analyzer can be used by Debug Tool to debug assembler programs, and you can create a readable listing from a Fault Analyzer side file or a SYSDEBUG file generated by the COBOL compiler.

You can also use the Interactive System Productivity Facility (ISPF) panels in Debug Tool to invoke File Manager Base, DB2 or IMS functions and a user exit enables you to specify a TEST run-time option string in the DB2, IMS or batch environments.

Debug in many environments

IBM Debug Tool can help you debug an application while it runs in a host environment, such as a batch application, TSO, ISPF, CICS, IMS or DB2 (including IBM DB2 stored procedures) environments. Debug Tool can help you debug almost any application and almost any host language, including COBOL, PL/I, C/C++ and Assembler applications running on z/OS systems.

With Debug Tool, you can compile and link your COBOL, PL/I, C and C++ programs, and assemble and link Assembler programs — as well as pre-process and compile your CICS and DB2 programs.

IBM Rational Developer for System z works with Debug Tool, to give your developers a fully integrated development, test and debugging environment for all applications running on z/OS, whether traditional, SOA or Web-based.

A CICS utility transaction (CADP or DTCN) enables you to control debugging in the CICS environment. For example, you can debug based on a specific program or transaction name, while other CICS-specific capabilities enable you to specify the span of a debug session or view — or edit CICS storage and diagnose storage violations.

Display and alteration of 64-bit general purpose registers in assembler expressions is provided on hardware that supports 64-bit addressing. Debug Tool correctly displays data items according to type, including three floating-point data types: binary (IEEE), decimal and hexadecimal.

New in V11.1

Debug Tool for z/OS, V11.1 includes several new additions and enhancements, including:

- *A plug-in, for use with Eclipse-based platforms like CICS Explorer, provides a GUI interface to the host Debug Tool product. It is available as a web download for licensed customers of Debug Tool.*

- *A new mode of operation, explicit debug mode, is now supported. In this mode, the user identifies the compile units to debug, then Debug Tool loads debug data only for those compile units. This mode can significantly improve debugger performance when it is debugging very large and complex programs. This new mode is an alternative to the standard Debug Tool mode of operation where debug data is automatically loaded for all compile units. It is intended to be used only when debugging large, complex applications that don't perform as well in the standard Debug Tool mode.*
- *A new user interface is added to the Terminal Interface Manager (TIM) that helps you create and manage the TEST runtime options data set.*
- *The TIM has been enhanced to remove the need for a site to set up a separate TN3270E port or to customize a set of terminal LUs.*
- *A GUI interface is added that helps you create and manage the TEST runtime options data set from the workstation.*
- *The Debug Tool Language Environment user exit for DB2 (EQADDCXT) now supports debugging of DB2 stored procedures of type SUB invoked using the call_sub function.*
- *EQAOPTS commands can now be specified at run-time in addition to the use of a user-generated EQAOPTS load module. This allows individual users to enter EQAOPTS commands at run-time by supplying a data set containing EQAOPTS commands.*
- *Support is added for debugging of assembler programs that exploit the latest IBM zEnterprise 196 architecture.*
- *Enhanced performance while debugging C and C++ applications.*
- *A Popup window, which displays the result of the LIST expression command when the Log window is not visible.*
- *Support for the Enterprise PL/I ADDRDATA built-in function.*
- *Support for the Enterprise PL/I V4.1 compiler and its new GONUMBER(SEPARATE) option.*
- *For programs compiled with any level of Enterprise PL/I, you can now list a single element of an array of structures. For programs compiled with Enterprise PL/I V4.1, you can list a single element of an array of structures in automonitor or use the L prefix command in the Source window to list a single element of an array of structures.*
- *For programs compiled with any level of Enterprise PL/I, you can now change the format in which Debug Tool displays an array. By using the SET LIST BY SUBSCRIPT ON command, you can have Debug Tool display the array as it is stored in memory.*
- *A new keyword LABELS is added to the LIST NAMES command where you can list the names of all section and paragraph names in a COBOL program, and the names of all instruction labels in an assembler program.*
- *The following breakpoints are enhanced:*
    - o *AT CHANGE and AT LABEL breakpoints are enhanced to allow a user to limit the scope of the breakpoint to a specific compile unit.*
    - o *AT GLOBAL is enhanced to provide an OCCURRENCE option or wild card (*) to stop for any condition raised in the application.*
    - o *The QUERY LOCATION command is enhanced to provide more information when Debug Tool stops for an AT CHANGE breakpoint.*
- *Automonitor enhancements:*
    - o *You can change the subscripts of an array directly in the Monitor window.*
    - o *You can delete multiple items from Monitor window at one time.*
    - o *You can use the cursor (in combination with the CLEAR MONITOR command) to indicate which variable to remove from the Monitor window.*
- *Automated allocation of the commands, log, preferences, save settings and save breakpoints and monitor specifications files.*
- *New functions are included in Debug Tool Utilities to help an application programmer more easily start debugging IMS applications running in BTS.*
- *A CICS transaction, DTNP, is provided which issues NEWCOPY or PHASEIN of application programs.*
- *Documentation is provided to assist debugging of Language Environment C/C++, COBOL, and PLI programs in the Java JNI environment in z/OS.*

IBM Debug Tool for z/OS, V11.1, at a glance
Hardware requirements
• Any hardware configuration capable of running IBM z/OS
Software requirements
• IBM z/OS, V1.10 (5694-A01) or later • Depending on the functions used in the product, other related products may be required. See the product Web site, listed below, for more detailed software requirements

Part of a leading-edge family of z/OS tools

Debug Tool for z/OS is part of the IBM Problem Determination family of products. These include IBM Application Performance Analyzer for z/OS, IBM Debug Tool for z/OS, IBM Fault Analyzer for z/OS, IBM File Manager for z/OS, and IBM Workload Simulator for OS/390 and z/OS.

Designed to help you maximize your investment in IBM System z products, these products are a robust suite of integrated development tools that enable you to improve IT operational efficiency and transform applications to achieve greater business flexibility.

The IBM Problem Determination tools help application developers to improve application delivery throughout the application life cycle. You can use these tools to help increase productivity and IT effectiveness across source code debugging, application ABEND analysis, data management and application performance analysis.

The IBM Problem Determination tools do much more than support traditional applications. They include capabilities that enable you to build SOA applications. They are tightly integrated with other tools in the IBM problem determination tools portfolio, as well as other tool sets. The tools also continue to support and make the most of the latest subsystem levels. These capabilities help make IBM problem determination tools an excellent choice for your business.

For more information

To learn more about IBM problem determination tools, contact your IBM representative or IBM Business Partner, or visit the following Web sites:

IBM Problem Determination Tools family:
ibm.com/software/awdtools/deployment/

IBM Application Performance Analyzer for z/OS:
ibm.com/software/awdtools/apa/

IBM Debug Tool for z/OS:
ibm.com/software/awdtools/debug tool/

IBM Fault Analyzer for z/OS:
ibm.com/software/awdtools/faultanalyzer

IBM File Manager for z/OS:
ibm.com/software/awdtools/filemanager

IBM Workload Simulator for z/OS and OS/390:
ibm.com/software/awdtools/workloadsimulator

TAKE BACK CONTROL WITH WebSphere.