

IBM Debug Tool for z/OS, V12.1

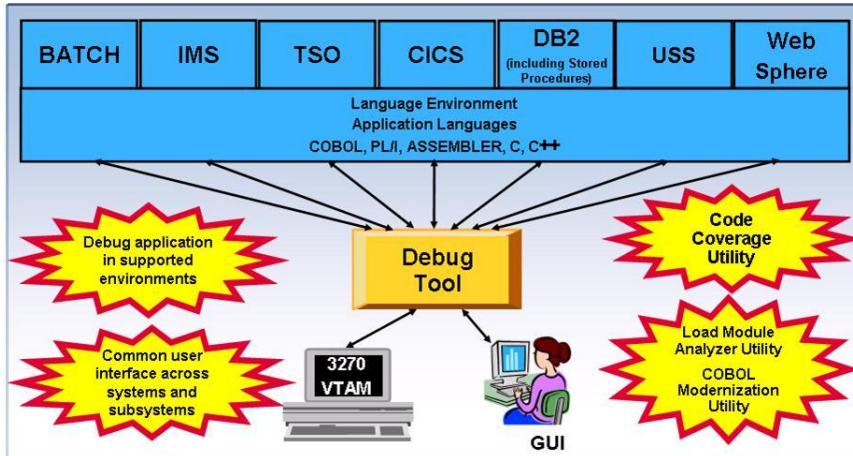


Figure 1: Debug Tool environment

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**
- **Workstation GUI interface**

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®, V12.1 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, V12.1 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, V12.1 product includes all of the function in the previous separate products as well as the new V12.1 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.

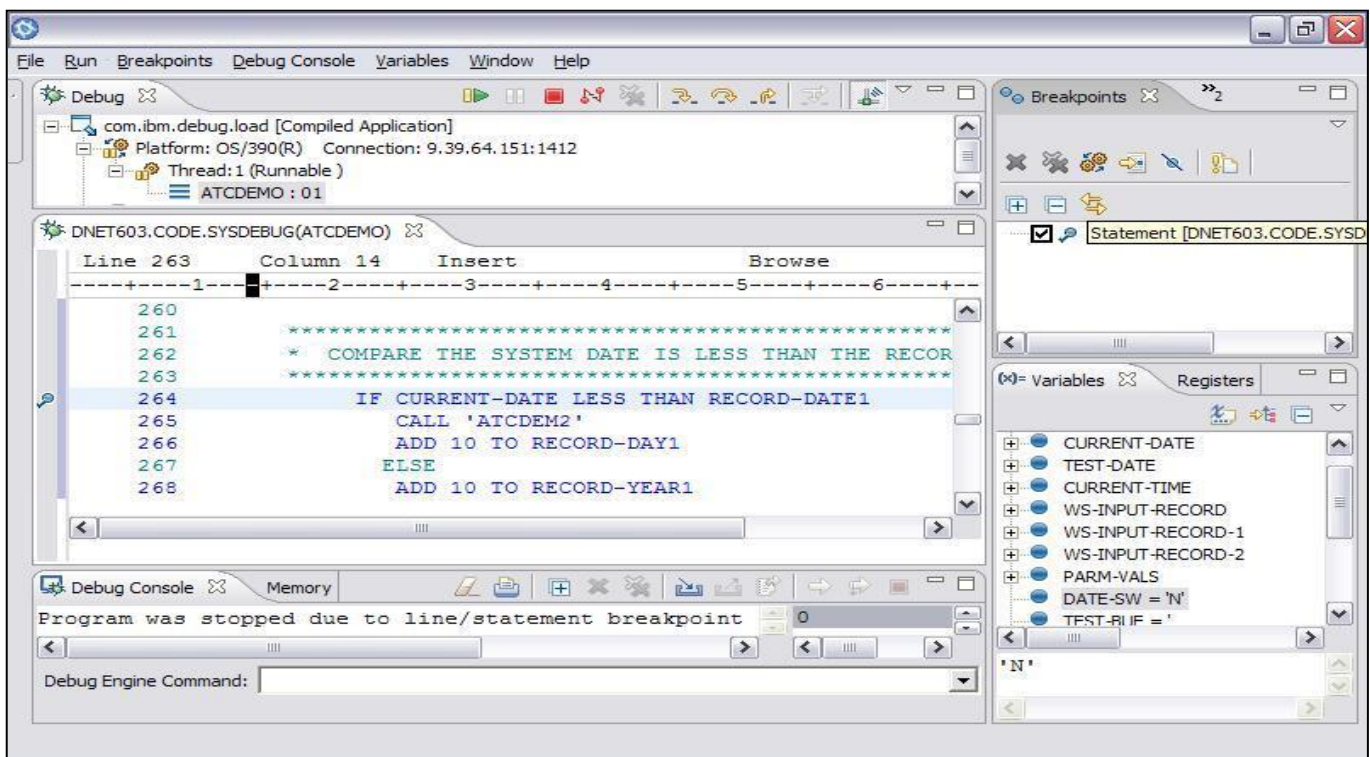


Figure 2: Rational Developer for System z works with Debug Tool to help mainframe developers be more productive.

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

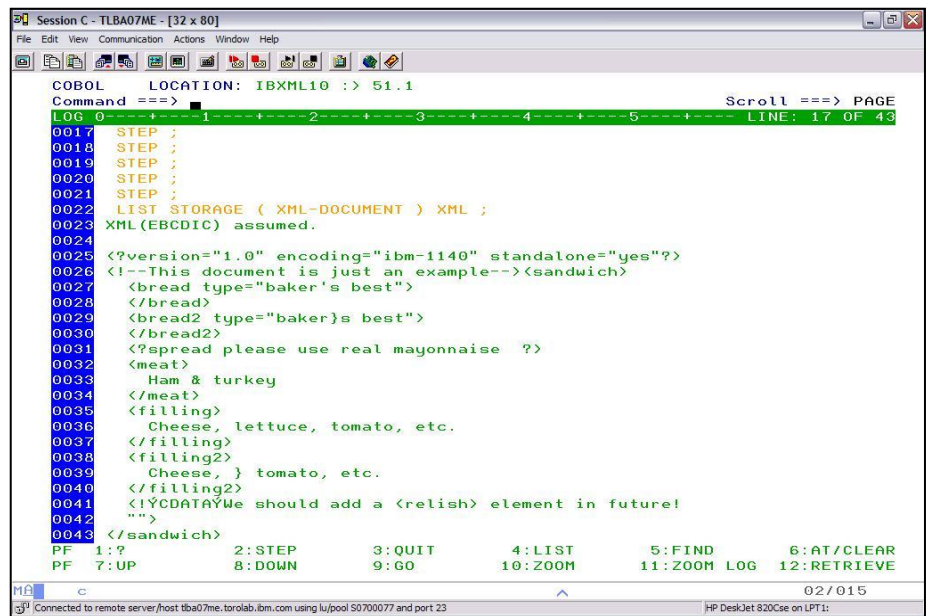
Previously, to create faster, more efficient applications, you had to sacrifice debugging support. With Debug Tool you can debug Enterprise COBOL for z/OS applications that have been compiled with standard or full-optimization compiler options.

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.

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.



```
COBOL LOCATION: IBXML10 :> 51.1
Command ==>
LOG 0 1 2 3 4 5
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 <!--YCDATAWe 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
02 / 015
Connected to remote server/host tba07me.torolab.ibm.com using lu/pool 50700077 and port 23
HP DeskJet 820Cse on LPT1:
```

Figure 3: Formatted XML structure using List Storage command

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 V12.1

- Support for the IBM Explorer for z/OS component in CICS Explorer V1.1.1 and IMS Enterprise Suite Explorer V2.1.2. IBM Explorer for z/OS provides a common framework for managing connections between multiple plug-ins and their server components. When using the framework, users are prompted only once per session for their ID and password, even when accessing multiple tools on the same LPAR. IBM Explorer for z/OS also includes a perspective, views, and APIs to list, create, edit, and control z/OS datasets, partitioned datasets, zFS files and paths and to submit jobs and view JES spool output.
- Delay debug mode is added, enabling Debug Tool to operate in a dormant state until a compile unit of interest to the user becomes active. Delay debug mode is valid for non-CICS programs written in Enterprise PL/I or Enterprise COBOL. When delay debug is on, Debug Tool delays starting the initialization for the debugging session until a compile unit matching the program naming pattern in the delay debug profile is found. That matched compile unit is started using the TEST runtime options in the delay debug profile.
- You can now debug non-CICS C and C++ applications that use XPLINK linkage without having to insert compiled-in hooks. Additional benefits include smaller load modules, enhanced performance, and improved interaction between Java-based applications and COBOL, PL/I, and Assembler.
- The three Debug Tool Language Environment user exits (EQADBCXT, EQADDCXT, and EQADICXT) are merged into a single one: EQAD3CXT.
- Remote communication security is enhanced to enable encrypted communication between Debug Tool and remote debug mode for better privacy and integrity. The enablement uses the Secure Sockets Level (SSL) protocol and digital certificates.
- DTCN enhancement prevents a non-owner from deleting a profile. The 3270 BMS DTCN transaction checks to see if users are authorized to delete or modify other owners' profiles, and rejects unauthorized attempts.
- DTCN profile plug-in is integrated with CICS objects in the CICS Explorer. This provides a new mechanism to create or manage a DTCN profile when you want to debug a transaction or program.
- The Coverage Utility increases the number of monitor sessions from 32 to 256.
- You can specify a SET LIST BY SUBSCRIPT command to control the display format when you view a COBOL array. You can change it to display as it is stored in memory. This enhancement is for remote debug mode only.
- Previously, Debug Tool allowed duplicate monitor commands to be entered. Now users can prevent duplicate monitor definitions. This enhancement potentially saves space in the monitor window and in the monitor commands list.
- Debug Tool does not establish a Monitor command for Monitor Local List expression if one or more variables are not defined in the specified compile unit. This enhancement reduces unnecessary information in the monitor window.
- Storage overhead is reduced, and performance of automonitor is improved when you debug a program with large structures.
- PL/I conditions associated with file handling can now use a wildcard instead of users having to specify a file reference.
- The EQAOPTS DYNDEBUG command is added so that you can specify an initial default for the SET DYNDEBUG command.
- The %CHAR Debug Tool built-in function is added to enable you to display the result of an expression in EBCDIC format.
- The NONLESP option is added to EQANMDBG to enable you to direct Debug Tool to use a different storage subpool for its storage in cases where the program being debugged does a FREEMAIN of subpool 1 (where Debug Tool places its data by default).
- A new parameter, COMPOPTS, is now supported by the load module analyzer. COMPOPTS lists the compiler options that are known at runtime to be listed for each CSECT in the load module.
- The new EQAOPTS QUIET keyword for the SVCSCREEN,OFF command can be used to suppress the EQA2458I message.
- The new EQAOPTS STARTSTOPMSG command tells Debug Tool whether to issue a message when each Debug Tool session starts and stops.
- The Debug Tool Utilities enable you to customize the site default data set name for new users for the options Debug Tool User Exit Data Set and Delay Debug Profile.
- The Terminal Interface Manager now allows you to customize the site default data set name for the option "LE options data set".
- The LOADDEBUGDATA command has been enhanced to support %CU and %PROGRAM substitution for the cu_name operand.
- POSIX multi-process support is enhanced for remote debug mode.

IBM Debug Tool for z/OS, V12.1, at a glance

Hardware requirements

- Any hardware configuration capable of running IBM z/OS

Software requirements

- IBM z/OS, V1.11 (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



© Copyright IBM Corporation 2012

IBM Corporation
555 Bailey Ave
San Jose, CA 95141, U.S.A.

Produced in the U.S.A.
April 2012
All Rights Reserved

IBM, the IBM logo, ibm.com, and WebSphere are trademarks of IBM Corporation, registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at ibm.com/legal/copytrade.shtml.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc, in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation, in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries

Other company, product and service names may be trademarks or service marks of others.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates.

The IBM home page on the internet can be found at ibm.com

TAKE BACK CONTROL WITH **WebSphere.**