FRONT

Advanced, interactive source-level debugging for your C, COBOL, and PL/I applications

Debug Tool for VSE/ESA

1.0 One tool to debug them all!

Debug Tool for VSE/ESA (Debug Tool) provides advanced facilities for debugging and testing applications compiled with any combination of the following IBM high-level language compilers:

- IBM C for VSE/ESA
- IBM COBOL for VSE/ESA
- IBM PL/I for VSE/ESA

You can debug applications interactively using a full-screen interface or in batch mode using a predefined command file.

You can single-step through your application, or Debug Tool can be dynamically invoked when an error condition occurs--for example, an error that would normally cause an abend. Debug Tool displays a source-level view of the point of failure, and provides facilities for diagnosing and correcting the problem. You can make adjustments and re-execute code that previously failed without leaving the debug session. You do not need to analyze dumps to determine which statement failed.

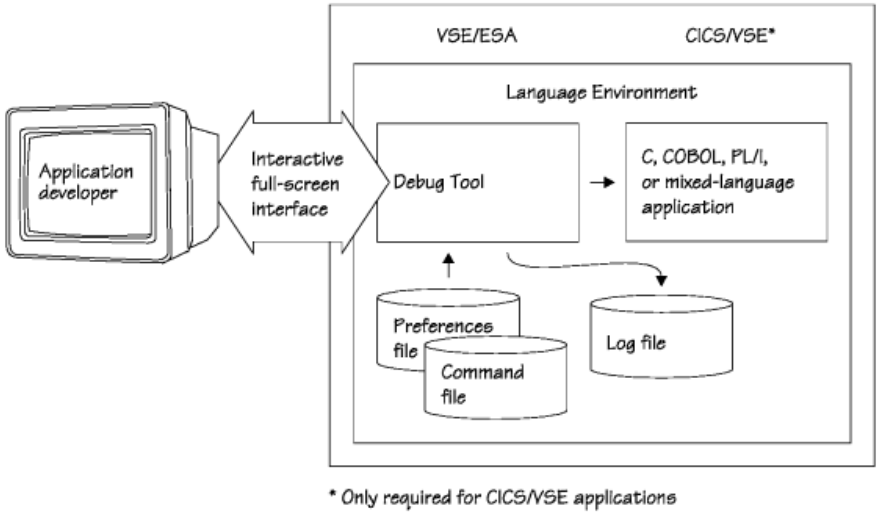

Figure 1. Debugging your application using Debug Tool

```
 ____________________________________________________________________________
|                                                                            |
| COBOL    LOCATION: COBPROG :> 100.1                                        |
| Command ===>                                              Scroll ===> PAGE |
| MONITOR --+----1----+----2----+----3----+----4----+----5----+----6 LINE: 1 OF 3 |
| ****************************** TOP OF MONITOR ******************************  | |
| 0001  1 77 COBPROG:>VARBL2    21                                           | | 1
| 0002  2 77 COBPROG:>VARBL1    11                                           | |
| 0003  3 77 COBPROG:>X    1                                                 | |
| ***************************** BOTTOM OF MONITOR *****************************  | |
| SOURCE: COBPROG --1----+----2----+----3----+----4----+----5---- LINE: 98 OF 118 |
|       98              ADD 1 TO VARBL1                                    .  | |
|       99              ADD 1 TO VARBL2                                    .  | | 2
|      100              CALL "SUBPRO1" USING BY CONTENT PARAM1              .  | |
|      101              ADD 1 TO X                                         .  | |
|      102            END-PERFORM.                                         .  | |
| LOG 0----+----1----+----2----+----3----+----4----+----5----+---- LINE: 13 OF 19 |
| 0013  STEP ;                                                               | |
| 0014  MONITOR                                                              | | 3
| 0015    LIST VARBL2 ;                                                      | |
| 0016  MONITOR                                                              | |
| 0017    LIST VARBL1 ;                                                      | |
| 0018  MONITOR                                                              | |
| 0019    LIST X ;                                                           | |
| 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 |
|                                                                            |
|____________________________________________________________________________|
```

1 **Monitor window:** Displays status of items you choose to monitor, such as: variables, registers, programs, the execution environment, and Debug Tool settings. For example, you can use this window to watch the content of variables change during program execution.

2 **Source window:** Displays the program source, with the current statement highlighted. In the prefix area at the left of this window, you can enter commands to set, display, and remove breakpoints.

3 **Log window:** Records and displays your interactions with Debug Tool and, optionally, shows program output. This window contains the same information as the log file (see Figure 1).

Figure 2. Debug Tool full-screen interface

2.0 Debug Tool features

Interactive or batch debugging

Using the full-screen interface, you can interactively debug any application as it runs (including batch applications). You can invoke Debug Tool when an application is initialized, or it can be started dynamically when a condition occurs. The application itself can also invoke Debug Tool. Figure 2 in topic 1.0 shows a sample display from the full-screen interface.

You can also debug your batch applications with Debug Tool in batch mode, using a predefined command file.

Source-level debug

You can monitor source code, for C, or a full compiler listing, for COBOL and PL/I. (Compiler listings are an expanded source listing that--in addition to program source--contain copy book source, for COBOL, or include file source, for PL/I.)

Debug Tool provides a COBOL and PL/I compiler exit to write compiler listings to permanent disk files--rather than the spool queue--making compiler listings available for debugging.

Debug mixed-language applications

Debug Tool supports seamless debugging of mixed-language applications within the same session, and recognizes when the current programming language changes (see "Dynamic patching" below). Language modules not supported by Debug Tool, such as assembler, are tolerated, but no debugging support is provided for these modules.

Dynamic patching

For each supported programming language, there is a set of interpreted commands that you can use to specify actions to be taken. These commands are subsets of the programming languages, so they are easy to learn, and allow you to make adjustments to your application while you debug it.

You can use the commands to alter the value of variables and structures, and to control the flow of an application. For example, a programmer can declare a new variable and use the variable to patch a program as it executes.

Dynamic breakpoints

You can set breakpoints in an application program, monitor variables for changes, and watch for specified exceptions and conditions during program execution. For example, you can cause an application to break when a specific variable or location in storage is changed. You can set, change, and remove breakpoints as you go through the application. You do not need to know where you want to break before you start.

Single-step debugging

To focus on a problem area, you can step line by line through the execution of an application. For example, when an application stops for a breakpoint, you can carefully examine each line that follows. Single-step debugging, along with the ability to set dynamic breakpoints, allows you to monitor, interrupt, and continue through the flow of the program to identify errors easily.

Program frequency

Debug Tool counts how many times a statement or verb has been processed in an application program. This allows you to verify the coverage of your code paths.

Program information

Debug Tool can display program and environment information. You can display, monitor, and alter program variables or storage in real time. You can also check how your application was compiled, and look at its structure.

Session logging

Each debug session can be recorded in a log file for reviewing, editing, or replaying. This allows you to replay the actions taken in a session to pinpoint errors in an application.

Testing tool

You can also use Debug Tool as a test tool. By using the session logging feature as you debug code, you can save the results of your session for use as input to a future Debug Tool session. As you make changes to your code, you can use the saved log file as input to Debug Tool in order to verify that no unexpected behavior occurs as a result of these changes. Session logging allows you to create suites of regression testcases that you can use to minimize the number of new bugs introduced during the normal application development process.

3.0 Starting a debug session

To invoke Debug Tool, you need to compile your application with the TEST compile-time option, link-edit it, then run it with the TEST run-time option:

```
    .
    .
    .
  *  Compile with TEST option
  *  EQALIST is the exit that writes the compiler
  *  listing to disk for later use by Debug Tool
  // EXEC IGYCRCTL,SIZE=IGYCRCTL,
                PARM='EXIT(PRTEXIT(''MYLIB.MYSUB'',EQALIST))'
  CBL TEST(ALL) RMODE(ANY)
    .
    .
    .
  COBOL source
    .
    .
    .
  /*
  *  Link-edit
  // EXEC PGM=LNKEDT
  /*
  *  Run with TEST option
  *  This example invokes the full-screen interface
  *  with the program loaded, waiting for commands
  // EXEC PGM=COBPROG,,PARM='/TEST(,,,MFI%D0820001:)'
    .
    .
    .
```

Figure 3. Example job control to invoke Debug Tool

In the example run step above, "MFI%D0820001:" specifies a 3270 VTAM terminal ID (LU name) that you have established. When the program runs,

this TEST run-time option causes the Debug Tool full-screen interface to be invoked, and a screen similar to Figure 2 in topic 1.0 is displayed on the specified 3270 terminal.

For CICS/VSE applications, you can invoke Debug Tool in one of two ways:

- Single-terminal mode: A single 3270 session for Debug Tool and the application; when the application is running, Debug Tool is hidden, and vice versa.
- Dual-terminal mode: One 3270 session is used to display the application, and the other is used for Debug Tool.

The application itself can also invoke Debug Tool:

- By specifying suboptions on the TEST run-time option, you can cause your application to invoke Debug Tool when it terminates normally, abends, or generates errors or conditions above a chosen severity.
- By using a library service call--such as CEETEST, PLITEST, or the ctest() function--your application can invoke Debug Tool directly.

4.0 Debug Tool and C run-time

Rather than requiring an additional run-time library, Debug Tool takes advantage of the Language Environment base and C run-time libraries provided with:

- VSE/ESA Version 2 Release 2, with integrated VSE C Language Run-Time Support installed
- IBM Language Environment for VSE/ESA Release 4, with C run-time library installed

Debug Tool requires that at least one of the above is installed on your system.

5.0 How to order Debug Tool

Debug Tool is not available as a separate product, but is packaged with the Full Function Offerings of the following licensed compiler programs:

- IBM C for VSE/ESA Release 1
- IBM COBOL for VSE/ESA Release 1
- IBM PL/I for VSE/ESA Release 1

Debug Tool is the same for each compiler; you need only order Debug Tool with one of these products.

6.0 How to learn more

If you'd like to learn more about Debug Tool and related products, visit the Debug Tool web site at http://www.software.ibm.com/ad/dtvse/ or call:

Australia	132 426
Austria	0222 21145 2500
Belgium	02 225 33 33
Canada	1 800 565 SW4U
Denmark	80 30 45 45
Finland	90 459 4837
France	36 63 36 43
Germany	0130 4567
Ireland	01 6603744
Italy	16 70 17 001
Netherlands	06 0220402
New Zealand	0800 800 809
Norway	66 99 93 00
Portugal	353 1 7915 111
South Africa	27 11 22 49 111
Spain	900 100 400
Sweden	08 793 40 04
Switzerland	155 1225
United Kingdom	01329 242728
United States	1 800 IBM 2468 (ext. STAR703)

Or contact your local IBM office. We look forward to hearing from you.

IBM

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service.

Any other documentation with respect to this licensed program, including any documentation referenced herein, is provided for reference purposes only and does not extend or modify these specifications.

Printed in U.S.A.

IBM Library Server Print Preview

DOCNUM = GC26-8925-00
DATETIME = 10/29/96 03:16:11
BLDVERS = 1.2
TITLE = Debug Tool/VSE V1R1 Fact Sheet
AUTHOR =

PATH = /home/webapps/epubs/htdocs/book
