Debug Tool for z/OS

Reference Summary

Version 10.1
This edition applies to Debug Tool for z/OS, Version 10.1 (Program Number 5655-V50), which supports the following compilers:

- AD/Cycle® C/370™ Version 1 Release 2 (Program Number 5688-216)
- C/C++ for MVS/ESA Version 3 (Program Number 5655-121)
- C/C++ feature of OS/390® (Program Number 5647-A01)
- C/C++ feature of z/OS (Program Number 5694-A01)
- OS/VS COBOL, Version 1 Release 2.4 (5740-CB1) - with limitations
- VS COBOL II Version 1 Release 3 and Version 1 Release 4 (Program Numbers 5668-958, 5688-023) - with limitations
- COBOL/370 Version 1 Release 1 (Program Number 5688-197)
- COBOL for MVS & VM Version 1 Release 2 (Program Number 5688-197)
- COBOL for OS/390 & VM Version 2 (Program Number 5648-A25)
- Enterprise COBOL for z/OS and OS/390 Version 4.2 and earlier (Program Number 5655-S71)
- High Level Assembler for MVS & VM & VSE Version 1 Release 4, Version 1 Release 5, and Version 1 Release 6 (Program Number 5696-234)
- PL/I for MVS & VM Version 1 Release 1 (Program Number 5688-235)
- VisualAge® PL/I for OS/390 Version 2 Release 2 (Program Number 5655-B22)
- Enterprise PL/I for z/OS and OS/390 Version 3.9 or earlier (Program Number 5655-H31)

This edition also applies to all subsequent releases and modifications until otherwise indicated in new editions or technical newsletters.

You can order publications online at www.ibm.com/shop/publications/order, or order by phone or fax. IBM Software Manufacturing Solutions takes publication orders between 8:30 a.m. and 7:00 p.m. Eastern Standard Time (EST). The phone number is (800)879-2755. The fax number is (800)445-9269.

You can find out more about Debug Tool by visiting the IBM Web site for Debug Tool at: http://www.ibm.com/software/awdtools/debugtool

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Contents

About this document .................................... vii
Who might use this document .............................. vii
Accessing z/OS licensed documents on the Internet vii
Using LookAt to look up message explanations .......... viii
How to read syntax diagrams ................................ viii
Symbols .................................................. ix
Syntax items ............................................. ix
Syntax examples .......................................... ix
How to send your comments ................................ x

Summary of changes ...................................... xiii
Changes introduced with Debug Tool V10.1 ........... xiii

Chapter 1. Debug Tool commands ........................ 1
? command ................................................ 1
ALLOCATE command ..................................... 1
ANALYZE command (PL/I) ............................. 1
Assignment command (assembler and disassembly) .... 1
Assignment command (non-Language Environment COBOL) .................. 2
Assignment command (PL/I) ............................ 2
AT ALLOCATE (PL/I) .................................. 2
AT APPEARANCE ....................................... 2
AT CALL ................................................. 2
AT CHANGE ............................................. 3
AT CHANGE (remote) ................................... 3
AT CURSOR (full-screen mode) .......................... 3
AT DATE (COBOL) ..................................... 3
AT DELETE ............................................. 4
AT ENTRY ............................................... 4
AT ENTRY (remote) ..................................... 4
AT EXIT ................................................ 4
AT GLOBAL ............................................ 4
AT LABEL .............................................. 5
AT LINE ............................................... 5
AT LOAD ............................................... 5
AT LOAD (remote) ..................................... 5
AT OCCURRENCE ...................................... 6
AT OFFSET (disassembly) ................................ 6
AT PATH ............................................... 6
AT Prefix (full-screen mode) ............................ 6
AT STATEMENT ........................................ 6
AT STATEMENT (remote) ................................ 7
AT TERMINATION ...................................... 7
BEGIN command ........................................ 7
block command (C and C++) ............................ 7
break command (C and C++) ............................ 7
CALL %CEBR .......................................... 7
CALL %CECI .......................................... 8
CALL %DUMP .......................................... 8
CALL %FA ............................................ 8
CALL %FM ............................................ 8
CALL %HOGAN ....................................... 8
CALL %VER .......................................... 8
CALL entry_name (COBOL) ............................. 9
CALL procedure ........................................ 9
CHKSTGV .............................................. 9
CLEAR command ....................................... 9
CLEAR prefix (full-screen mode) ........................ 10
COMMENT command .................................... 10
COMPUTE command (COBOL) ........................... 11
CURSOR command (full-screen mode) ................... 11
Declarations (assembler, disassembly, and non-Language Environment COBOL) .................. 11
Declarations (C and C++) ................................ 11
Declarations (COBOL) .................................. 13
DECLARE command (PL/I) ............................. 13
DESCRIBE command .................................... 15
DISABLE command ...................................... 15
DISABLE prefix (full-screen mode) ........................ 16
DO command (assembler, disassembly, and non-Language Environment COBOL) .................. 16
DO command (PL/I) .................................... 16
do/while command (C and C++) ........................ 17
ENABLE command ...................................... 17
ENABLE prefix (full-screen mode) ........................ 17
EVALUATE command (COBOL) .......................... 17
Expression command (C and C++) ........................ 18
FIND command ........................................ 18
FINDBP command ...................................... 18
for command (C and C++) ................................ 19
FREE command ......................................... 19
GO command .......................................... 19
GOTO command ....................................... 19
GOTO LABEL command ................................ 19
%IF command (programming language neutral) ........ 20
IF command (assembler, disassembly, and non-Language Environment COBOL) .................. 20
if command (C and C++) ................................ 20
IF command (COBOL) .................................. 20
IF command (PL/I) .................................... 20
IMMEDIATE command (full-screen mode) ................ 20
INPUT command (C and C++ and COBOL) ............. 21
JUMPTO command ...................................... 21
JUMPTO LABEL command ............................... 21
LIST (blank) ......................................... 21
LIST AT .............................................. 21
LIST CALLS ......................................... 22
LIST CONTAINER ..................................... 22
LIST CURSOR (full-screen mode) ........................ 23
LIST DTCN or CADP ................................... 23
LIST expression ....................................... 23
L expression prefix .................................... 23
LIST FREQUENCY ..................................... 24
LIST LAST ............................................ 24
LIST LINE NUMBERS .................................. 24
LIST LINES ........................................... 24
LIST MONITOR ........................................ 24
LIST NAMES .......................................... 24
LIST ON (PL/I) ...................................... 25
LIST PROCEDURES .................................... 25

© Copyright IBM Corp. 1992, 2009
BROWSE ............................................. 56
CACHENUM ............................................. 56
CODEPAGE ............................................. 56
DEFAULTVIEW ............................................ 56
DTCNFORCE xxx xxx .................................. 56
EQAQPP ................................................. 57
GPFD SN .................................................. 57
MDBG ...................................................... 57
NAMES ..................................................... 57
NODISPLAY .............................................. 57
SAVEBPDSN and SAVESETDSN ......................... 57
SUBSYS ................................................... 58
SVCS SCREEN ............................................ 58
THREADTERMCOND ..................................... 58
TIMACB ................................................... 58

Notices ................................................... 59
Copyright license ....................................... 60
Programming interface information ................... 60
Trademarks and service marks ......................... 60
About this document

Debug Tool combines the richness of the z/OS® environment with the power of Language Environment® to provide a debugger for programmers to isolate and fix their program bugs and test their applications. Debug Tool gives you the capability of testing programs in batch, using a nonprogrammable terminal in full-screen mode, or using a workstation interface to remotely debug your programs.

This document contains a summary of commands, built-in functions, and EQAOPTS options provided by Debug Tool. Each topic contains the name of the command, built-in function, or EQAOPTS option and then a syntax diagram. For more information about each command or built-in function, refer to Debug Tool Reference and Messages. For more information about each EQAOPTS option, see Debug Tool Customization Guide.

Who might use this document

This document is intended for programmers using Debug Tool to debug high-level languages (HLLs) with Language Environment and assembler programs either with or without Language Environment. Throughout this document, the HLLs are referred to as C, C++, COBOL, and PL/I.

Debug Tool runs on the z/OS operating system and supports the following subsystems:

- CICS®
- DB2®
- IMS™
- JES batch
- TSO
- UNIX® System Services in remote debug mode or full-screen mode using a dedicated terminal only
- WebSphere® in remote debug mode or full-screen mode using a dedicated terminal only

To use this document and debug a program written in one of the supported languages, you need to know how to write, compile, and run such a program.

Accessing z/OS licensed documents on the Internet

z/OS licensed documentation is available on the Internet in PDF format at the IBM® Resource Link™ Web site at:

http://www.ibm.com/servers/resourcelink

Licensed documents are available only to customers with a z/OS license. Access to these documents requires an IBM Resource Link user ID and password, and a key code. With your z/OS order you received a Memo to Licensees, (GI10-0671), that includes this key code.

To obtain your IBM Resource Link user ID and password, log on to:

http://www.ibm.com/servers/resourcelink

To register for access to the z/OS licensed documents:
1. Sign in to Resource Link using your Resource Link user ID and password.
2. Select User Profiles located on the left-hand navigation bar.

Note: You cannot access the z/OS licensed documents unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

Printed licensed documents are not available from IBM.

You can use the PDF format on either z/OS Licensed Product Library CD-ROM or IBM Resource Link to print licensed documents.

Using LookAt to look up message explanations

LookAt is an online facility that lets you look up explanations for most of the IBM messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can use LookAt from the following locations to find IBM message explanations for z/OS elements and features, z/VM®, VSE/ESA, and Clusters for ADX® and Linux®:

- Your z/OS TSO/E host system. You can install code on your z/OS or z/OS.e systems to access IBM message explanations, using LookAt from a TSO/E command line (for example, TSO/E prompt, ISPF, or z/OS UNIX System Services running OMVS).
- Your Microsoft® Windows® workstation. You can install code to access IBM message explanations on the z/OS Collection (SK3T-4269), using LookAt from a Microsoft Windows command prompt (also known as the DOS command line).
- Your wireless handheld device. You can use the LookAt Mobile Edition with a handheld device that has wireless access and an Internet browser (for example, Internet Explorer for Pocket PCs, Blazer, or Eudora for Palm OS, or Opera for Linux handheld devices). Link to the LookAt Mobile Edition from the LookAt Web site.

You can obtain code to install LookAt on your host system or Microsoft Windows workstation from a disk on your z/OS Collection (SK3T-4269), or from the LookAt Web site (click Download, and select the platform, release, collection, and location that suit your needs). More information is available in the LOOKAT.ME files available during the download process.

How to read syntax diagrams

This section describes how to read syntax diagrams. It defines syntax diagram symbols, items that may be contained within the diagrams (keywords, variables, delimiters, operators, fragment references, operands) and provides syntax examples that contain these items.

Syntax diagrams pictorially display the order and parts (options and arguments) that comprise a command statement. They are read from left to right and from top to bottom, following the main path of the horizontal line.
Symbols

The following symbols may be displayed in syntax diagrams:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>/SM590000/SM590000</td>
<td>Indicates the beginning of the syntax diagram.</td>
</tr>
<tr>
<td>──/SM590000</td>
<td>Indicates that the syntax diagram is continued to the next line.</td>
</tr>
<tr>
<td>/SM590000───</td>
<td>Indicates that the syntax is continued from the previous line.</td>
</tr>
<tr>
<td>──/SM590000/SM630000</td>
<td>Indicates the end of the syntax diagram.</td>
</tr>
</tbody>
</table>

Syntax items

Syntax diagrams contain many different items. Syntax items include:

- Keywords - a command name or any other literal information.
- Variables - variables are italicized, appear in lowercase and represent the name of values you can supply.
- Delimiters - delimiters indicate the start or end of keywords, variables, or operators. For example, a left parenthesis is a delimiter.
- Operators - operators include add (+), subtract (-), multiply (*), divide (/), equal (=), and other mathematical operations that may need to be performed.
- Fragment references - a part of a syntax diagram, separated from the diagram to show greater detail.
- Separators - a separator separates keywords, variables or operators. For example, a comma (,) is a separator.

Keywords, variables, and operators may be displayed as required, optional, or default. Fragments, separators, and delimiters may be displayed as required or optional.

<table>
<thead>
<tr>
<th>Item type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required</td>
<td>Required items are displayed on the main path of the horizontal line.</td>
</tr>
<tr>
<td>Optional</td>
<td>Optional items are displayed below the main path of the horizontal line.</td>
</tr>
<tr>
<td>Default</td>
<td>Default items are displayed above the main path of the horizontal line.</td>
</tr>
</tbody>
</table>

Syntax examples

The following table provides syntax examples.

<table>
<thead>
<tr>
<th>Item</th>
<th>Syntax example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required item.</td>
<td>Require_item.</td>
</tr>
<tr>
<td>Required items appear on the main path of the horizontal line. You must specify these items.</td>
<td>Require_item.</td>
</tr>
<tr>
<td>Required choice.</td>
<td>Require_choice1 Require_choice2.</td>
</tr>
<tr>
<td>A required choice (two or more items) appears in a vertical stack on the main path of the horizontal line. You must choose one of the items in the stack.</td>
<td>Require_choice1 Require_choice2.</td>
</tr>
</tbody>
</table>
Table 1. Syntax examples (continued)

<table>
<thead>
<tr>
<th>Item</th>
<th>Syntax example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional item.</td>
<td></td>
</tr>
<tr>
<td>Optional items appear below the main path of the horizontal line.</td>
<td><img src="Image" alt="optional_item" /></td>
</tr>
<tr>
<td>Optional choice.</td>
<td></td>
</tr>
<tr>
<td>An optional choice (two or more items) appears in a vertical stack below the main path of the horizontal line. You may choose one of the items in the stack.</td>
<td><img src="Image" alt="optional_choice1" />, <img src="Image" alt="optional_choice2" /></td>
</tr>
<tr>
<td>Default.</td>
<td></td>
</tr>
<tr>
<td>Default items appear above the main path of the horizontal line. The remaining items (required or optional) appear on (required) or below (optional) the main path of the horizontal line. The following example displays a default with optional items.</td>
<td><img src="Image" alt="default_choice1" />, <img src="Image" alt="optional_choice2" />, <img src="Image" alt="optional_choice3" /></td>
</tr>
<tr>
<td>Variable.</td>
<td></td>
</tr>
<tr>
<td>Variables appear in lowercase italics. They represent names or values.</td>
<td><img src="Image" alt="variable" /></td>
</tr>
<tr>
<td>Repeatable item.</td>
<td></td>
</tr>
<tr>
<td>An arrow returning to the left above the main path of the horizontal line indicates an item that can be repeated.</td>
<td><img src="Image" alt="repeatable_item" /></td>
</tr>
<tr>
<td>A character within the arrow means you must separate repeated items with that character.</td>
<td><img src="Image" alt="repeatable_item" />,</td>
</tr>
<tr>
<td>An arrow returning to the left above a group of repeatable items indicates that one of the items can be selected, or a single item can be repeated.</td>
<td><img src="Image" alt="repeatable_item" /></td>
</tr>
<tr>
<td>Fragment.</td>
<td></td>
</tr>
<tr>
<td>The │ fragment │ symbol indicates that a labelled group is described below the main syntax diagram. Syntax is occasionally broken into fragments if the inclusion of the fragment would overly complicate the main syntax diagram.</td>
<td><img src="Image" alt="fragment" /></td>
</tr>
</tbody>
</table>

How to send your comments

Your feedback is important in helping us to provide accurate, high-quality information. If you have comments about this document or any other Debug Tool documentation, contact us in one of these ways:

- Use the Online Readers’ Comment Form at www.ibm.com/software/awdtools/rcf/. Be sure to include the name of the document, the publication number of the document, the version of Debug Tool, and, if applicable, the specific location (for example, page number) of the text that you are commenting on.
- Fill out the Readers’ Comment Form at the back of this document, and return it by mail or give it to an IBM representative. If the form has been removed, address your comments to:
IBM Corporation  
H150/090  
555 Bailey Avenue  
San Jose, CA 95141-1003  
USA  

- Fax your comments to this U.S. number: (800)426-7773.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.
Summary of changes

This section lists the key changes made to Debug Tool for z/OS.

Changes introduced with Debug Tool V10.1

The following changes, if applicable, are marked with revision bars:

- A RESTful HTTP access interface to read, create, update, and delete profiles in the DTTCN profile repository has been added. The interface is described in Debug Tool API User’s Guide and Reference. An example of a GUI interface that uses the RESTful HTTP access interface to manipulate the profiles from the workstation is also available. How to download and install this example is described in Debug Tool API User’s Guide and Reference.

- If you are using z/OS XLC C/C++, Version 1 Release 11, Debug Tool has added support for .mdbg files that contain source. The .mdbg file contains the debug information and a copy of the source needed in your debug session. You no longer need to have access to the source while debugging your program.

If you are not familiar with .mdbg files and how to create them, see the topics "dbgld - Create a module map for debugging" or "CDADBGLD - Create a debug side file for the module map" in z/OS XL C/C++ User’s Guide.

You can indicate that Debug Tool always use .mdbg files to search for source and debug information by setting the EQAXOFT option MDBG to YES in the EQAOPTS options file. To learn how to set the MDBG option, see “Specifying whether Debug Tool searches through .mdbg files” in Debug Tool Customization Guide. The syntax diagram in Chapter 3, “EQAOPTS options,” on page 55 has been updated to include the MDBG option. In situations where you can specify environment variables, you can set the environment variable EQA_USE_MDBG to YES or NO, which overrides any setting (including the default setting) of the EQAXOPT MDBG option.

To learn what compiler options to choose to create .dbg files, which the dbglld command or CDADBGLD utility use to build the .mdbg files, see the following topics in Debug Tool User’s Guide:

- “Choosing DEBUG compiler options for C programs”
- “Choosing DEBUG compiler options for C++ programs”

There are several different methods of specifying the location of .dbg and .mdbg files. The following list summarizes each method:

- While you are debugging your program, you can use the following commands:
  - “SET DEFAULT DBG” on page 38
  - “SET DEFAULT MDBG” on page 39
  - Only for full screen, batch, and line mode: “SET MDBG (C, C++)” on page 43

After you specify the location of the .dbg and .mdbg files, you can use the following commands to verify the location:

- The QUERY SET DEFAULT DBG and QUERY SET DEFAULT MDBG commands, which are described in “QUERY command” on page 32.
- Only for full screen, batch, and line mode: QUERY SET MDBG, which is described in “QUERY command” on page 32.
All of these commands are described in Debug Tool Reference and Messages.

- In your JCL, you can add EQADBG and EQAMDBG DD statements and specify the data set name of the corresponding .dbg or .mdbg file. To learn more about using these DD statements, see “Compiling your program without using Debug Tool Utilities” in Debug Tool User’s Guide. If you are debugging in UNIX System Services on in CICS, you cannot use these DD statements.

- In UNIX System Services, you can use the following environment variables:
  - EQA_DBG_PATH
  - EQA_MDBG_PATH

- The term full-screen mode through a VTAM® terminal has been changed to full-screen mode using a dedicated terminal. The term was changed to remove the implication that any instructions that referred to VTAM terminal applied only to those terminals connected through an SNA network.

- You can now use IBM Session Manager while debugging in full-screen mode using a dedicated terminal and using the Terminal Interface Manager. The following topics have been added to or updated in the Debug Tool Customization Guide:
  - “Example: a debugging session using the Debug Tool Terminal Interface Manager”
  - “Starting the Debug Tool Terminal Interface Manager”
  - “Configuring Terminal Interface Manager as an IBM Session Manager application”

- In the Debug Tool User’s Guide, the topic “Starting a debugging session in full-screen mode using a dedicated terminal” has been updated.

- The CODEPAGE(ccsid) option has been added to the XML option of the LIST CONTAINER and LIST STORAGE commands to improve the display of character strings encoded in an alternate code page on a 3270 terminal.

- You can now add all the variables in the Working-Storage Section of a COBOL program to the Monitor window with one command. In Debug Tool User’s Guide, the topic “Displaying the Working-Storage Section of a COBOL program in the Monitor window” has been added to describe how to add these variables to the Monitor window.

- In Debug Tool Reference and Messages, the topic “MONITOR command” has been updated to describe the new suboption WSS.

- In Debug Tool Reference Summary, the syntax diagram in “MONITOR command” on page 27 has been updated to include the new suboption WSS.

- In full screen mode, a new window called the Command pop-up window has been added that makes it easier to enter and edit long commands. In Debug Tool User’s Guide, the following topics have been added or updated:
  - “Command pop-up window”
  - “Opening the Command pop-up window to enter long Debug Tool commands”
  - “Entering multiline commands in full-screen”
In **Debug Tool Reference Summary** and **Debug Tool Reference and Messages**, the following topics have been added:

- “POPUP command” on page 31
- “SET POPUP” on page 44

You can now enter changes to multiple variables in the Monitor window at one time.

In **Debug Tool User’s Guide**, an item on the list in “Restrictions for modifying variables in the Monitor window” has been removed.

Debug Tool now supports automatic saving and restoring of breakpoints and settings for IMS Transaction Manager (TM) programs.

In **Debug Tool User’s Guide**, phrases that describe this limitation have been removed from the following topics:

- “Restoring Manually”
- “Data sets used by Debug Tool”

In **Debug Tool Reference and Messages**, phrases that describe this limitation have been removed from the following topics:

- “SET RESTORE command”
- “SET SAVE command”

In **Debug Tool Reference and Messages**, the syntax diagram for “LIST expression” on page 23 has been updated to include the use of the GROUP option for COBOL programs.

Two commands, POSITION and FINDBP, have been added to improve the ability to scroll to a specific line. You can use POSITION integer, which is similar to SCROLL TO integer, to scroll to a particular line or statement. FINDBP, which is similar to FIND, searches for line, statement, or offset breakpoints in the Source window.

In **Debug Tool User’s Guide**, the following topics have been updated to clarify how to scroll to a particular line:

- “Scrolling to a particular line number”
- “Displaying the line at which execution halted”

In **Debug Tool Reference and Messages** and **Debug Tool Reference Summary**, the commands FINDBP and POSITION are described in topics “FINDBP command” on page 18 and “POSITION command” on page 31.

The way to identify, in DTCN profiles, the program you want to debug has changed.

Previously, you identified a program through the Program ID field. This has changed to two fields: LoadMod and CU.

In **Debug Tool Reference Summary**, the following syntax diagrams have been updated to describe the new options:

- “DISABLE command”
- “ENABLE command”
- “LIST DTCN or CADP command”
- DTCNFORCELOADMODID, which is described in “EQAOPTS options”
- DTCNFORCECUID, which is described in “EQAOPTS options”

In **Debug Tool Reference and Messages**, the following syntax diagrams have been updated to describe the new options:

- “DISABLE command”
- “ENABLE command”
- “LIST DTCN or CADP command”
In Debug Tool User’s Guide, the instructions in “Creating and storing a DTCN profile” have been updated to describe the new fields.

In Debug Tool Customization Guide, DTCNFORCELOADMODID and DTCNFORCECUID have been added to “Defining EQAOPTS options: checklist and instructions”.

- Debug Tool now supports running in browse mode. In this mode, you cannot make modifications to storage or registers, nor modify the control flow of a program with commands like GOTO and JUMPTO. You can debug a program, but you cannot change the behavior of a program. This might be useful when you want to debug a program running in a production environment but you want to prevent unauthorized changes to a program’s behavior or production data.

In Debug Tool Reference Summary and Debug Tool Reference and Messages, the description of the following commands have been updated to describe how you cannot use them in browse mode:

- “ALLOCATE command” on page 1
- “Assignment command (assembler and disassembly)” on page 1
- “Assignment command (non-Language Environment COBOL)” on page 2
- “Assignment command (PL/I)” on page 2
- “CALL %CECI” on page 8
- “CALL entry_name (COBOL)” on page 9
- “CALL %FM” on page 8
- “CALL %HOGAN” on page 8
- CLEAR LOG command, which is described in “CLEAR command” on page 9
- “COMPUTE command (COBOL)” on page 11
- “FREE command” on page 19
- G0 BYPASS command, which is described in “GO command” on page 19
- “GOTO command” on page 19
- “GOTO LABEL command” on page 19
- “INPUT command (C and C++ and COBOL)” on page 21
- “JUMPTO command” on page 21
- “JUMPTO LABEL command” on page 21
- “MEMORY” on page 27
- “MOVE command (COBOL)” on page 27
- “QUIT command” on page 34
- QUIT expression command, which is described in “QUIT command” on page 34
- “QQUIT command” on page 34
- “SET INTERCEPT (C and C++)” on page 41
- “SET INTERCEPT (COBOL, full-screen mode, line mode, batch mode)” on page 42
- “SET INTERCEPT (COBOL, remote debug mode)” on page 42
- “SET command (COBOL)” on page 47
- “STORAGE command” on page 48
- “SYSTEM command” on page 49
- “TRIGGER command” on page 49
- “TSO command (z/OS)” on page 50
The topic “QUERY command” on page 32 has been updated to describe the new option BROWSE MODE.

In Debug Tool User’s Guide, the topic “Choosing a debugging mode” has been updated to describe how browse mode works and how you control browse mode.

In Debug Tool Customization Guide, the following topics have been added to describe the customization tasks you must do for this feature:

- “Installing the browse mode RACF® facility”
- “Enabling users to control browse mode”

• Debug Tool now supports displaying more than 1000 lines in the Monitor window.

In Debug Tool Reference and Messages, the usage note that describes this limitation has been removed from “MONITOR command.”

In Debug Tool Reference and Messages and Debug Tool Reference Summary, the following topics have been updated:

- The topic “MONITOR command” on page 27 has been updated to describe the new option LIMIT.
- The topic “QUERY command” on page 32 has been updated to describe the new option MONITOR LIMIT.

In the Debug Tool User’s Guide, the topic “Monitor window” has been updated to describe how to increase the number of lines that the Monitor window displays and the implications of monitoring large volumes of data.

• Debug Tool now supports monitoring, by using the AT CHANGE command, of assembler variables with dynamically updated addresses such as those in a DSECT.

In Debug Tool Reference and Messages, an existing usage note has been modified and a new usage note has been added to “AT CHANGE (full screen mode, line mode, batch mode)” that describes how Debug Tool monitors these variables.

• Debug Tool now supports debugging C and C++ programs that run in the Airline Control System (ALCS). This support is available only if you debug in remote debug mode.

In the Debug Tool User’s Guide, the following topics have been updated:

- “A table that lists the supported subsystems” has been updated to indicate that Debug Tool supports the ALCS subsystem.
- “Choosing TEST or NOTEST compiler suboptions for C programs” has been updated to indicate that if you want to debug C and C++ programs running in ALCS, you must compile them with hooks.
- “Choosing TEST or NOTEST compiler options for C++ programs” has been updated to indicate that if you want to debug C and C++ programs running in ALCS, you must compile them with hooks.
- “Choosing a debugging mode” has been updated to indicate for the ALCS subsystem, you must choose remote debug mode.

• Debug Tool now supports using the L and M prefix commands for assembler and disassembly programs.

In the Debug Tool User’s Guide, the following topics have been updated to describe how to use the L and M prefix commands on assembler and disassembly programs:

- “Displaying the value of a variable”
- “Entering prefix commands on specific lines or statements”
- “Displaying and monitoring the value of a variable”
- “One-time display of the value of variables”
- “Adding variables to the Monitor window”

Some of these topics also describe a slight change in terminology. These topics use the word “operand” to mean a variable in C, C++, COBOL, or PL/I, or the operand of an assembler instruction.

In Debug Tool Reference and Messages, the topics “L prefix command (full-screen mode)” and “M prefix (full-screen mode)” have been updated to describe how to identify operands or variables on a statement, describe the limitations of this support, and show a new example.

- In the Debug Tool User’s Guide, the topic “Quick Start guide for compiling and assembling programs for use with IBM Problem Determination Tools products” which helps you choose the compiler options that work for all Problem Determination Tools, has been added.

- In Debug Tool Customization Guide, all of the EQAOPTS options have been organized into one topic: “Defining EQAOPTS options: checklist and instructions” This will help you keep track of the changes you are making to EQAOPTS so that you can make those changes at one time.
Chapter 1. Debug Tool commands

Debug Tool provides the following commands:

? command

Displays a list of all commands or, if used in combination with a command, displays a list of options that you can specify for that command.

ALLOCATE command

The ALLOCATE command allocates a file (ddname) to an existing data set, a concatenation of existing data sets, or a temporary data set.

attributes:

ANALYZE command (PL/I)

The ANALYZE command displays the process of evaluating an expression and the data attributes of any intermediate results.

Assignment command (assembler and disassembly)

The Assignment command copies the value of an expression to a specified memory location or register.
Assignment command (non-Language Environment COBOL)

The Assignment command assigns the value of an expression to a specified reference. It is the equivalent of the COBOL COMPUTE statement.

```
receiver = sourceexpr;
```

Assignment command (PL/I)

The Assignment command copies the value of an expression to a specified reference.

```
reference = expression;
```

AT ALLOCATE (PL/I)

AT ALLOCATE gives Debug Tool control when storage for a named controlled variable or aggregate is dynamically allocated by PL/I.

```
AT every_clause ALLOCATE identifier, command;
```

AT APPEARANCE

Gives Debug Tool control when the specified compile unit is found in storage.

```
AT every_clause APPEARANCE cu_spec, command;
```

AT CALL

Gives Debug Tool control when the application code attempts to call the specified entry point.

```
AT every_clause CALL entry_name, command;
```
AT CHANGE

Gives Debug Tool control when either the program or Debug Tool command changes the specified variable value or storage location, and, optionally, when the specified condition is met.

```
AT every_clause CHANGE

reference WHEN condition
%STORAGE (address, length)

command ;
```

AT CHANGE (remote)

Gives Debug Tool control when either the program or Debug Tool command changes the specified variable value.

```
AT—CHANGE "reference" ;
```

AT CURSOR (full-screen mode)

Provides a cursor controlled method for setting a statement breakpoint.

```
AT TOGGLE CURSOR ;
```

AT DATE (COBOL)

Gives Debug Tool control for each date processing statement within the specified block.

```
AT every_clause DATE block_spec command ;
```
AT DELETE

Gives Debug Tool control when a load module is removed from storage by a Language Environment delete service, such as on completion of a successful C release(), COBOL CANCEL, or PL/I RELEASE.

```plaintext
AT every_clause DELETE load_spec command;
```

AT ENTRY

Defines a breakpoint at the specified entry point in the specified block.

```plaintext
AT every_clause ENTRY block_spec WHEN condition command;
```

AT ENTRY (remote)

Defines a breakpoint at the specified entry point in the specified block.

```plaintext
AT ENTRY block_spec;
```

AT EXIT

Defines a breakpoint at the specified exit point in the specified block.

```plaintext
AT every_clause EXIT block_spec command;
```

AT GLOBAL

Gives Debug Tool control for every instance of the specified AT-condition.

```plaintext
AT GLOBAL
```
AT LABEL

Gives Debug Tool control when execution has reached the specified statement label or group of labels.

```
AT every_clause LABEL statement_label command;
```

AT LINE

Gives Debug Tool control at the specified line. See "AT STATEMENT" on page 6.

AT LOAD

Gives Debug Tool control when the specified load module is brought into storage.

```
AT every_clause LOAD module_name command;
```

AT LOAD (remote)

Gives Debug Tool control when the specified load module is brought into storage.

```
AT LOAD module_name;
```
**AT OCCURRENCE**

Gives Debug Tool control on a language or Language Environment condition or exception.

```
AT every_clause OCCURRENCE condition command ;
```

**AT OFFSET (disassembly)**

Gives Debug Tool control at the specified offset in the disassembly view.

```
AT OFFSET offset_spec command ;
```

**AT PATH**

Gives Debug Tool control when the flow of control changes (at a path point). `AT PATH` is identical to `AT GLOBAL PATH`.

```
AT every_clause PATH command ;
```

**AT Prefix (full-screen mode)**

Sets a statement breakpoint when you issue this command through the Source window prefix area.

```
AT integer ;
```

**AT STATEMENT**

Gives Debug Tool control at each specified statement or line within the given set of ranges, and, optionally, when the specified condition is met.

```
AT every_clause LINE statement_id_range command ;
WHEN condition command ;
```
AT STATEMENT (remote)

Gives Debug Tool control at each specified statement or line.

```
AT STATEMENT statement_id;
```

AT TERMINATION

Gives Debug Tool control when the application program is terminated.

```
AT TERMINATION command;
```

BEGIN command

BEGIN and END delimit a sequence of one or more commands to form one longer command.

```
BEGIN command END;
```

block command (C and C++)

The block command allows you to group any number of Debug Tool commands into one command.

```
{ command };
```

break command (C and C++)

The break command allows you to terminate and exit a loop (that is, do, for, and while) or switch command from any point other than the logical end.

```
break;
```

CALL %CEBR

Starts the CICS Temporary Storage Browser Program.

```
CALL %CEBR;
```
CALL %CECI

Starts the CICS Command Level Interpreter Program.

CALL %DUMP

Calls the Language Environment dump service to obtain a formatted dump.

CALL %FA

Starts and instructs IBM Fault Analyzer to provide a formatted dump of the current machine state.

CALL %FM

Starts IBM File Manager for z/OS.

CALL %HOGAN

Starts Computer Sciences Corporation’s KORE-HOGAN application, also known as SMART (System Memory Access Retrieval Tool).

CALL %VER

Adds a line to the log describing the maintenance level of Debug Tool that you have installed on your system.
CALL entry_name (COBOL)

Calls an entry name in the application program.

CALL identifier literal USING identifier_clause;

identifier_clause:

CALL procedure

Calls a procedure that has been defined with the PROCEDURE command.

CALL procedure_name;

CHKSTGV

Check for a specific type of storage violation in the task you are currently debugging.

CHKSTGV;

CLEAR command

The CLEAR command removes the actions of previously issued Debug Tool commands.
CLEAR prefix (full-screen mode)

Clears a breakpoint when you issue this command through the Source window prefix area.

>>> CLEAR \[integer\];

COMMENT command

The COMMENT command can be used to insert commentary into the session log.

>>> COMMENT \[commentary\];
**COMPUTE command (COBOL)**

The **COMPUTE** command assigns the value of an arithmetic expression to a specified reference.

```
COMPUTE reference = expression;
```

**CURSOR command (full-screen mode)**

The **CURSOR** command moves the cursor between the last saved position on the Debug Tool session panel (excluding the header fields) and the command line.

```
CURSOR;
```

**Declarations (assembler, disassembly, and non-Language Environment COBOL)**

Use declarations to create session variables and tags effective during a Debug Tool session.

```
identifier – DS – F
               /Ln – X, XLn – C, CLn – H, HLn – A, ALn – B, BLn – P, PLn – Z, ZLn – E, D / L
```

**Declarations (C and C++)**

Use declarations to create session variables and tags effective during a Debug Tool session.

```
scalar_def | enum_def | struct_def | union_def | declarator
```

Chapter 1. Debug Tool commands 11
Declarations (COBOL)

Use declarations to create session variables effective during a Debug Tool session.

attribute:

usage_attribute:

DECLARE command (PL/I)

The DECLARE command creates session variables effective during a Debug Tool session.
major_structure:

```
│                │
├───────────────┼───────────────
│ level–name   │ attribute     
└───────────────┴───────────────
```

scalar:

```
│                │
├───────────────┼───────────────
│ name          │ attribute     
│ name          │ attribute     
└───────────────┴───────────────
```

14 Debug Tool V10.1 Reference Summary
**DESCRIBE command**

The **DESCRIBE** command displays the file allocations or attributes of references, compile units, known load modules, and the run-time environment.

```
CURSOR
  DESCRIPTER
    ALLOCATIONS
      USER
      ALL
      SYSTEM
      LINKLIST
      LPALIST
      APF LIST
      CATALOG
      PARMLIB
      PROCLIB

ATTRIBUTES
  reference
  "reference"
  (reference)

CHANNEL
  *
  channel_name

CUS
  PROGRAMS
    cu_spec
    (cu_spec)

ENVIRONMENT
  LOADMODS
    load_spec
    (load_spec)
```

**DISABLE command**

The **DISABLE** command makes an AT breakpoint inoperative or prevents Debug Tool from being started by CADP or DTCN. However, the breakpoint is not cleared. Later, you can make the breakpoint operative or allow Debug Tool to be started by CADP or DTCN by using the **ENABLE** command.

```
DISABLE
  AT_command
    CADP *
      PROGRAM prog_id
      CU cu_id
    DTCN *
      LOADMOD loadmod_id
      CU cu_id
```
**DISABLE prefix (full-screen mode)**

Disables a statement breakpoint or offset breakpoint when you issue this command through the Source window prefix area.

```
DISABLE integer;
```

**DO command (assembler, disassembly, and non-Language Environment COBOL)**

The `DO` command performs one or more commands that are collected into a group.

```
DO ; command ; END ;
```

**DO command (PL/I)**

The `DO` command allows one or more commands to be collected into a group that can (optionally) be repeatedly executed.

**Simple**

```
DO ; command ; END ;
```

**Repeating**

```
DO WHILE (expression) ; UNTIL (expression) ;
UNTIL (expression) ; WHILE (expression) ;
```

**Iterative**

```
DO reference ; iteration ; END ;
```
iteration:

- `expression`
- `BY expression`
- `TO expression`
- `REPEAT expression`

- `WHILE (expression)`
- `UNTIL (expression)`

**do/while command (C and C++)**

The do/while command performs a command before evaluating the test expression.

```plaintext
do-command while (expression); 
```

---

**ENABLE command**

The ENABLE command activates an AT or pattern match breakpoint after it was disabled.

```plaintext
ENABLE AT_command
```

**ENABLE prefix (full-screen mode)**

Enables a disabled statement breakpoint or a disabled offset breakpoint when you issue this command through the Source window prefix area.

```plaintext
ENABLE integer
```

---

**EVALUATE command (COBOL)**

The EVALUATE command provides a shorthand notation for a series of nested IF statements.
any_clause:

- ANY
- condition
- TRUE
- FALSE
- NOT
- constant
- reference
- THROUGH
- constant
- THRU
- reference

Expression command (C and C++)

The Expression command evaluates the given expression.

expression;

FIND command

The FIND command helps you search through the Source window in full-screen and batch mode, and through the Log and Monitor windows in full-screen mode.

FIND

string

leftcolumn

rightcolumn

FIRST
LAST
CURSOR
LOG
MONITOR
SOURCE

FINDBP command

The FINDBP command provides full-screen search capability for line, statement and offset breakpoints in the source object.
for command (C and C++)

The for command provides iterative looping similar to the C and C++ for statement.

\[
\text{for(expression; expression; expression)} \quad \text{command;}
\]

FREE command

The FREE command releases a file that is currently allocated.

\[
\text{FREE FILE ddname;}
\]

GO command

The GO command causes Debug Tool to start or resume running your program.

\[
\text{GO BYPASS;}
\]

GOTO command

The GOTO command causes Debug Tool to resume program execution at the specified statement id.

\[
\text{GOTO statement_id;}
\]

GOTO LABEL command

The GOTO LABEL command causes Debug Tool to resume program execution at the specified statement label. The specified label must be in the same block. If you want Debug Tool to return control to you at the target location, make sure there is a breakpoint at that location.

\[
\text{GOTO LABEL statement_label;}
\]
%IF command (programming language neutral)

The %IF command enables you write conditional statements that can be run in any supported programming language.

```
%IF condition THEN command ELSE command;
```

IF command (assembler, disassembly, and non-Language Environment COBOL)

The IF command lets you conditionally perform a command.

```
IF condition THEN command ELSE command;
```

if command (C and C++)

The if command lets you conditionally perform a command.

```
if (expression) command else command;
```

IF command (COBOL)

The IF command lets you conditionally perform a command.

```
IF condition THEN command END-IF;
```

IF command (PL/I)

The IF command lets you conditionally perform a command.

```
IF expression THEN command ELSE command;
```

IMMEDIATE command (full-screen mode)

The IMMEDIATE command causes a command within a command list to be performed immediately.

```
IMMEDIATE command;
```
**INPUT command (C and C++ and COBOL)**

The **INPUT** command provides data for an intercepted read and is valid only when there is a read pending for an intercepted file.

```
INPUT text;
```

**JUMPTO command**

The **JUMPTO** command moves the resume point to the specified statement but does not resume the program.

```
JUMPTO statement_id;
```

**JUMPTO LABEL command**

The **JUMPTO** command moves the resume point to the specified label but does not resume the program.

```
JUMPTO LABEL 'statement_label';;
```

**LIST (blank)**

Displays the Source Identification panel, where associations are made between source listings or source files shown in the source window and their program units.

**LIST AT**

Lists the currently defined breakpoints, including the action taken when the specified breakpoint is activated.
LIST CALLS

Displays the dynamic chain of active blocks.

LIST CALLS;

LIST CONTAINER

Displays the contents of a CICS container.

LIST CONTAINER (channel_name) container_name

(index)

(sub_string_start::sub_string_end)

(sub_string_start:::sub_string_length)
LIST CURSOR (full-screen mode)

Provides a cursor controlled method for displaying variables, structures, and arrays.

LIST

LIST DTCN or CADP

List the programs and compile units that were disabled by the DISABLE command.

LIST DTCN;

LIST expression

Displays values of expressions.

LIST expression TITLED 'expression'

UNTITLED

GROUP reference

(1)

Notes:

1 Only for COBOL.

L expression prefix

Entered through the prefix area of the Source window, displays the value of a variable or variables on that line.
LIST FREQUENCY

Lists statement execution counts.

```
LIST FREQUENCY statement_id_range; integer
```

LIST LAST

Displays a list of recent entries in the history table.

```
LIST LAST integer HISTORY LINES PATHS STATEMENTS;
```

LIST LINE NUMBERS

Equivalent to LIST STATEMENT NUMBERS.

LIST LINES

Equivalent to LIST STATEMENTS.

LIST MONITOR

Lists all or selected members of the current set of MONITOR commands.

```
LIST MONITOR integer - integer;
```

LIST NAMES

Lists the names of variables, programs, or Debug Tool procedures.
LIST ON (PL/I)

Lists the action (if any) currently defined for the specified PL/I conditions.

LIST ON pli_condition;

LIST PROCEDURES

Lists the commands contained in the specified Debug Tool PROCEDURE definitions.

LIST PROCEDURES name,

 LIST STATEMENT NUMBERS

Lists all statement or line numbers that are valid locations for an AT LINE or AT STATEMENT breakpoint.

LIST LINE STATEMENT NUMBERS block_spec,

LIST STATEMENT NUMBERS

Displays the current register contents.
LIST STATEMENTS

Lists one or more statements or lines from a file.

```plaintext
LIST LINE statement_id_range;
```

LIST STORAGE

Displays the contents of storage at a particular address in hex format.

```plaintext
LIST STORAGE (address reference reference', offset length

XML

EBCDIC |
| ASCII |
| CODEPAGE (ccsid)
```

LOAD command

Specifies to load the named module using MVS LOAD services, EXEC CICS LOAD, or the Language Environment enclave-level load service for debugging purposes.

```plaintext
LOAD module_name;
```

LOADDEBUGDATA (LDD)

The LOADDEBUGDATA command specifies that a compile unit is an assembler compile unit and loads the debug data that corresponds to that assembler compile unit.

```plaintext
LOADDEBUGDATA LDD load_module_name::> cu_name
```
MEMORY

Identifies an address in memory and then display the contents of memory at that location in the Memory window. The Memory window displays memory in dump format.

```
MEMORY address
reference
' reference '
simple_expression
```

M prefix command

The M prefix command, which you enter through the prefix area of the Source window, adds variables on that line to the Monitor window.

```
M
integer
integer
target
```

MONITOR command

The MONITOR command defines or redefines a command whose output is displayed in the monitor window (full-screen mode), terminal output (line mode), or log file (batch mode).

```
MONITOR GLOBAL
LOCAL
命令 cu_spec
integer
HEX
DEFAULT
```

MOVE command (COBOL)

The MOVE command transfers data from one area of storage to another.

```
MOVE reference TO reference
```

NAMES DISPLAY command

Use the NAMES DISPLAY command to indicate that you want a list of all the load modules or compile units that are currently excluded or included. If you do not specify the ALL parameter, only the names excluded by user commands appear in the list that is displayed. Names that Debug Tool excludes by default are not included in the list that is displayed.
**NAMES EXCLUDE command**

The NAMES EXCLUDE command enables you to indicate to Debug Tool the names of load modules or compile units that you do not need to debug. If these are data-only modules, Debug Tool does not process them. If they contain executable code, Debug Tool might process them in some cases. See "Optimizing the debugging of large applications" in the Debug Tool User’s Guide for more information about these situations.

**NAMES INCLUDE command**

Use the NAMES INCLUDE command to indicate to Debug Tool that your program is a user load module or compile unit, not a system program. See "Debugging user programs that use system prefix names" in Debug Tool User’s Guide for more information.

**Null command**

The Null command is a semicolon written where a command is expected.

**ON command (PL/I)**

The ON command establishes the actions to be executed when the specified PL/I condition is raised.
The PANEL command displays special panels.

**PERFORM command (COBOL)**

The PERFORM command transfers control explicitly to one or more statements and implicitly returns control to the next executable statement after execution of the specified statements is completed.

**Simple:**

```
PERFORM command END-PERFORM;
```

**Repeating:**

```
PERFORM command BEFORE TEST WITH AFTER END-PERFORM;
```
The `PLAYBACK BACKWARD` command indicates to Debug Tool to perform STEP and RUNTO commands backward, starting from the current point and going to previous points.

The `PLAYBACK DISABLE` command informs Debug Tool to stop recording run-time environment information and discard any information recorded thus far.

The `PLAYBACK ENABLE` command informs Debug Tool to begin recording the application run-time environment information (steps history and data history).

**options:**

```plaintext
<table>
<thead>
<tr>
<th>cuname</th>
<th>integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>NODATA</td>
<td></td>
</tr>
</tbody>
</table>
```
PLAYBACK FORWARD command

The PLAYBACK FORWARD command indicates to Debug Tool to perform STEP and RUNTO commands forward, starting from the current point and going to the next point.

```
PLAYBACK FORWARD;
```

PLAYBACK START command

The PLAYBACK START command suspends normal debugging and informs Debug Tool to replay the steps it recorded.

```
PLAYBACK START;
```

PLAYBACK STOP command

The PLAYBACK STOP command stops replaying recorded statements and resumes normal debugging at the point where the PLAYBACK START command was entered.

```
PLAYBACK STOP;
```

POPUP command

Displays the Command pop-up window, where you can type in multiline commands.

```
POPUP integer;
```

POSITION command

Positions the cursor to a specific line in the specified window.

```
POSITION integer CURSOR LOG MONITOR SOURCE;
```

Prefix commands (full-screen mode)

The Prefix commands apply only to source listing lines and are typed into the prefix area in the source window. For details, see the section corresponding to the command name.

The following table summarizes the forms of the Prefix commands.

<table>
<thead>
<tr>
<th>Prefix command (full-screen mode)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT Prefix (full-screen mode)</td>
<td>Defines a statement breakpoint through the Source window prefix area.</td>
</tr>
</tbody>
</table>
"CLEAR prefix (full-screen mode)" on page 10
Clears a breakpoint through the Source window prefix area.

"DISABLE prefix (full-screen mode)" on page 16
Disables a breakpoint through the Source window prefix area.

"ENABLE prefix (full-screen mode)" on page 17
Enables a disabled breakpoint through the Source window prefix area.

"LIST expression" on page 23
Displays the value of a variable or variables on that line.

"QUERY prefix (full-screen mode)" on page 34
Queries what statements have breakpoints through the Source window prefix area.

"RUNTO prefix command (full-screen mode)" on page 35
Runs the program to the location that the cursor or statement identifier indicate in the Source window prefix area.

"SHOW prefix command (full-screen mode)" on page 47
Specifies what relative statement or verb within the line is to have its frequency count shown in the suffix area.

**PROCEDURE command**

The **PROCEDURE** command allows the definition of a group of commands that can be accessed by using the **CALL** procedure command.

```
<<< name:=PROCEDURE; command END; >>>
```

**QUALIFY RESET**

This command is equivalent to **SET QUALIFY RESET**.

**QUERY command**

The **QUERY** command displays the current value of the specified Debug Tool setting, the current setting of all the Debug Tool settings, or the current location in the suspended program.

```
<<< QUERY >>>
```

```
| Attributes A through I | Attributes J through P | Attributes Q through Z |
```

**Attributes A through I:**
Attributes J through P:

- KEYS
- LDD
- LIST-TABULAR
- LOCATION
- LOG
- LOG-NUMBERS
- LONGCUNAME
- MDBG
- MONITOR
- COLUMN
- DATATYPE
- LIMIT
- NUMBERS
- WRAP
- MSGID
- LANGUAGE
- NATIONAL
- PACE
- PFKEYS
- PLAYBACK
- PLAYBACK-LOCATION
- POPUP
- PROGRAMMING
- LANGUAGE
- PROMPT

Attributes Q through Z:
Notes:
1. You can use this command in remote debug mode.
2. Available only if the Dynamic Debug facility is installed.
3. Only for PL/I.

**QUERY prefix (full-screen mode)**

Queries what statements on a particular line have statement breakpoints when you issue this command through the Source window prefix area.

```
--QUERY ;
```

**QUIT command**

The QUIT command ends a Debug Tool session and if an expression is specified, sets the return code.

```
--QUIT (expression) ;
```

**QQUIT command**

The QQUIT command ends a Debug Tool session without further prompting.

```
--QQUIT ;
```
RESTORE command

The RESTORE command enables you to explicitly restore the settings, breakpoints, and monitor specifications that were previously saved by the SET SAVE AUTO command when Debug Tool terminated.

```
>> RESTORE SETTINGS;
```

RETRIEVE command (full-screen mode)

The RETRIEVE command displays the last command entered on the command line.

```
>> RETRIEVE COMMAND;
```

RUN command

The RUN command is synonymous to the GO command.

RUNTO command

The RUNTO command runs your program to a valid executable statement without setting a breakpoint.

```
>> RUNTO statement_id;
```

RUNTO prefix command (full-screen mode)

Runs to the statement when you issue this command through the Source window prefix area.

SCROLL command (full-screen mode)

The SCROLL command provides horizontal and vertical scrolling in full-screen mode.

```
>> SCROLL DOWN LEFT CSR NEXT DATA RIGHT HALF UP integer MAX PAGE BOTTOM TO integer TOP CURSOR;
```
SELECT command (PL/I)

The SELECT command chooses one of a set of alternate commands.

```
SELECT (reference) ;
WHEN (expression) command
OTHERWISE command
END ;
```

SET ASSEMBLER ON/OFF

The SET ASSEMBLER ON/OFF command displays additional information that is useful when you debug an assembler program.

```
SET ASSEMBLER ON ; OFF
```

SET ASSEMBLER STEPOVER

The SET ASSEMBLER STEPOVER command specifies how Debug Tool processes STEP OVER commands in assembler compile units.

```
SET ASSEMBLER STEPOVER EXTONLY EXTINT
```

SET AUTOMONITOR

Controls the monitoring of data items at the statement that Debug Tool runs next, ran most recently, or both.

```
SET AUTOMONITOR ON NOLOG CURRENT ; OFF LOG PREVIOUS BOTH
```
SET CHANGE

Controls the frequency of checking the AT CHANGE breakpoints.

SET COLOR (full-screen and line mode)

Provides control of the color, highlighting, and intensity attributes when the SCREEN setting is ON.

color_attributes:

UI_elements:
SET COUNTRY

Changes the current national country setting for the application program.

```
SET COUNTRY country_code;
```

SET DBCS

Controls whether shift-in and shift-out codes are interpreted on input and supplied on DBCS output.

```
SET DBCS ON OFF;
```

SET DEFAULT DBG

Defines a default partitioned data set DD name or DS name that Debug Tool searches through to locate the .dbg files.

```
SET DEFAULT DBG ddname dsn, (dsn);
```
**SET DEFAULT LISTINGS**

Defines a default partitioned data set DD name or DS name whose members are searched for program source, listings, or separate debug files.

```
SET DEFAULT LISTINGS
        ddname, dsn
```

**SET DEFAULT MDBG**

Defines a default partitioned data set DD name or DS name that Debug Tool searches through to locate the .mdbg files.

```
SET DEFAULT MDBG
        ddname, dsn
```

**SET DEFAULT SCROLL (full-screen mode)**

Sets the default scroll amount that is used when a SCROLL command is issued without the amount specified.

```
SET DEFAULT SCROLL
        CSR : DATA, HALF, integer, MAX, PAGE
```

**SET DEFAULT VIEW**

Controls the default view for assembler compile units.

```
SET DEFAULT VIEW
        STANDARD, NOMACGEN
```
SET DEFAULT WINDOW (full-screen mode)

Specifies what window is selected when a window referencing command (for example, FIND, SCROLL, or WINDOW) is issued without explicit window identification and the cursor is outside the window areas.

```
>> SET DEFAULT WINDOW LOG;  
    MEMORY  
    MONITOR  
    SOURCE  
```

SET DISASSEMBLY

Controls whether the disassembly view is displayed in the Source window.

```
>> SET DISASSEMBLY ON  
    OFF  
```

SET DYNDEBUG

Controls whether to activate the Dynamic Debug facility.

```
>> SET DYNDEBUG ON  
    OFF  
```

SET ECHO

Controls whether GO and STEP commands are recorded in the log window when they are not subcommands.

```
>> SET ECHO ON  
    OFF  
```

SET EQUATE

Equates a symbol to a string of characters.

```
>> SET EQUATE identifier = string;  
```

SET EXECUTE

Controls whether commands from all input sources are performed or just syntax checked (primarily for checking USE files).

```
>> SET EXECUTE ON  
    OFF  
```
SET FIND BOUNDS

Specifies the default left and right columns for a FIND command in the Source window and in line mode that does not specify any columns information.

\[ \text{SET FIND BOUNDS leftcolumn rightcolumn}; \]

SET FREQUENCY

Controls whether statement executions are counted.

\[ \text{SET FREQUENCY ON OFF cu_spec (cu_spec)}; \]

SET HISTORY

Specifies whether entries to Debug Tool are recorded in the history table and optionally adjusts the size of the table.

\[ \text{SET HISTORY ON OFF integer}; \]

SET IGNORELINK

Specifies that any new LINK level (nested enclave) is ignored while the setting is ON.

\[ \text{SET IGNORELINK ON OFF}; \]

SET INTERCEPT (C and C++)

Intercepts input to and output from specified files.

\[ \text{SET INTERCEPT FILE file_spec ON OFF}; \]

Chapter 1. Debug Tool commands 41
**SET INTERCEPT (COBOL, full-screen mode, line mode, batch mode)**

Intercepts input to and output from the console.

```
SET INTERCEPT CONSOLE ON
```

**SET INTERCEPT (COBOL, remote debug mode)**

Intercepts output from COBOL DISPLAY statements.

```
SET INTERCEPT ON
```

**SET KEYS (full-screen and line mode)**

Controls whether PF key definitions are displayed when the SCREEN setting is ON.

```
SET KEYS ON
```

**SET LDD**

Controls how debug data is loaded for assemblies containing multiple CSECTs.

```
SET LDD SINGLE
```

**SET LIST TABULAR**

Controls whether to format the output of the LIST command in a tabular format.

```
SET LIST TABULAR OFF
```

**SET LOG**

Controls whether each command that Debug Tool runs and the output of that command is stored in the log file. Defines (or redefines) the name and location of the file and whether the information is appended to an existing file or is written over existing information.

```
SET LOG OFF
```
SET LOG NUMBERS (full-screen and line mode)

Controls whether line numbers are shown in the log window.

```
SET LOG NUMBERS ON;  
SET LOG NUMBERS OFF;  
```

SET LONGCUNAME (C, C++, and PL/I)

Controls whether the CU name is displayed in short or long format.

```
SET LONGCUNAME ON;  
SET LONGCUNAME OFF;  
```

SET MDBG (C, C++)

Associates a .mdbg file to one load module or DLL.

```
SET MDBG (lm_spec) fileid;  
```

SET MONITOR (full-screen and line mode)

Controls the format and layout of variable names and values displayed in the Monitor window.

```
SET MONITOR COLUMN DATATYPE ON NUMBERS OFF WRAP LIMIT integer;  
```

SET MSGID

Controls whether the Debug Tool messages are displayed with the message prefix identifiers.

```
SET MSGID ON;  
SET MSGID OFF;  
```
SET NATIONAL LANGUAGE

Switches your application to a different run-time national language that determines what translation is used when a message is displayed.

```
/>=SET NATIONAL LANGUAGE-language_code;
```

SET PACE

Specifies the maximum speed (in steps per second) of animated execution.

```
/>=SET PACE-number;
```

SET PFKEY

 Associates a Debug Tool command with a Program Function key (PF key).

```
/>=SET PFn-string = command;
```

SET POPUP

Controls the number of lines displayed in the Command pop-up window.

```
/>=SET POPUP-integer;
```

SET PROGRAMMING LANGUAGE

Sets the current programming language.

```
/>=SET PROGRAMMING LANGUAGE CYCLE AUTOMATIC HOLD ASSEMBLER C COBOL DISASSEMBLY NONLCOBOL PLI;
```

SET PROMPT (full-screen and line mode)

Controls whether the current program location is automatically shown as part of the prompt message in line mode.

```
/>=SET PROMPT LONG SHORT;
```
SET QUALIFY
Simplifies the identification of references and statement numbers by resetting the point of view to a new block, compile unit, or load module.

```
SET QUALIFY
  BLOCK block_spec
    CU cu_spec
    PROGRAM address
    LOAD load_spec
  RESET
  RETURN
  UP
```

SET REFRESH (full-screen mode)
Controls screen refreshing.

```
SET REFRESH [ON|OFF]
```

SET RESTORE
Controls the restoring of settings, breakpoints, and monitor specifications.

```
SET RESTORE SETTINGS [NOAUTO]
  BPS AUTO
  MONITORS
```

SET REWRITE (full-screen mode)
Forces a periodic screen rewrite during long sequences of output.

```
SET REWRITE [EVERY] number
```

SET REWRITE (remote debug mode)
Sets the maximum number of COBOL DISPLAY statements that the remote debugger displays in the Debug Console.

```
SET REWRITE [EVERY] number
```

SET SAVE
Controls the saving of settings, breakpoints, and monitor specifications.
SET SCREEN (full-screen and line mode)

Controls how information is displayed on the screen.

SET SCROLL DISPLAY (full-screen mode)

Controls whether the scroll field is displayed when operating in full-screen mode.

SET SEQUENCE (PL/I)

Controls whether Debug Tool interprets data after column 72 in a commands or preference file as a sequence number.

SET SOURCE

 Associates a source file, compiler listing or separate debug file with one or more compile units.
SET SUFFIX (full-screen mode)
Controls the display of frequency counts at the right edge of the Source window when in full-screen mode.

```plaintext
SET SUFFIX [ON|OFF];
```

SET TEST
Overrides the initial TEST run-time options specified at invocation.

```plaintext
SET TEST [test_level|(-test_level-)];
```

SET WARNING (C, C++, and PL/I)
Controls display of the Debug Tool warning messages and whether exceptions are reflected to the application program.

```plaintext
SET WARNING [ON|OFF];
```

SET command (COBOL)
The SET command assigns a value to a COBOL reference.

```plaintext
SET [reference] TO [reference|literal|TRUE];
```

SHOW prefix command (full-screen mode)
The SHOW prefix command specifies what relative statement (for C) or relative verb (for COBOL) within the line is to have its frequency count temporarily shown in the suffix area.

```plaintext
SHOW [integer];
```
STEP command

The STEP command causes Debug Tool to dynamically step through a program, executing one or more program statements. In full-screen mode, it provides animated execution.

```
STEP integer INTO OVER RETURN;
```

STORAGE command

The STORAGE command enables you to alter up to eight bytes of storage.

```
STORAGE (address) = value;
```

switch command (C and C++)

The switch command enables you to transfer control to different commands within the switch body, depending on the value of the switch expression.

```
switch (expression) { switch_body };
```

**switch_body:**

```
switch_body: default_clause
                  |
                     |
                    |
                    |
                    |
```

**case_clause:**

```
case_expression:
  |
  |
  |
  |
  |
```

**default_clause:**

```
default:
  |
  |
  |
  |
  |
```

SYSTEM command

The SYSTEM command lets you issue TSO commands during a Debug Tool session.

```
> SYSTEM system_command;
```

TRIGGER command

The TRIGGER command raises the specified AT-condition in Debug Tool, or it raises the specified programming language condition in your program.

```
> TRIGGER AT condition;
```

Chapter 1. Debug Tool commands 49
storage_clause:

```
%^STORAGE(address, length)
```

---

**TSO command (z/OS)**

The TSO command lets you issue TSO commands during a Debug Tool session and is valid only in a TSO environment.

```
/TSM590000/SM590000 TSO tso_command;
```

---

**USE command**

The USE command causes the Debug Tool commands in the specified file or data set to be either performed or syntax checked.

```
/TSM590000/SM590000 USE ddname dsname;
```

---

**while command (C and C++)**

The while command enables you to repeatedly perform the body of a loop until the specified condition is no longer met or evaluates to false.

```
/TSM590000/SM590000 while (expression) command;
```

---

**WINDOW CLOSE**

Closes the specified window in the Debug Tool full-screen session panel.

```
/TSM590000/SM590000 WINDOW CLOSE CURSOR LOG MEMORY MONITOR SOURCE;
```

---

**WINDOW OPEN**

Opens a previously-closed window in the Debug Tool full-screen session panel.

```
/TSM590000/SM590000 WINDOW OPEN CURSOR LOG MEMORY MONITOR SOURCE;
```
WINDOW SIZE

Controls the relative size of currently visible windows in the Debug Tool full-screen session panel.

WINDOW SWAP

Replaces the logical window being displayed in a physical window with another logical window. The order of the operands is not important.

WINDOW ZOOM

Expands the indicated window to fill the entire screen or restores the screen to the currently defined window configuration.
Chapter 2. Debug Tool built-in functions

Debug Tool provides you with the following built-in functions:

%DEC (assembler, disassembly, and non-Language Environment COBOL)
Returns the decimal value of an operand.

%GENERATION (PL/I)
Returns a specific generation of a controlled variable in your program.

%HEX
Returns the hexadecimal value of an operand.

%INSTANCES (C, C++, and PL/I)
Returns the maximum value of %RECURSION (the most recent recursion number) for a given block.

%RECURSION (C, C++, and PL/I)
Returns a specific instance of an automatic variable or a parameter in a recursive procedure.

%WHERE (assembler, disassembly, and non-Language Environment COBOL)
Returns a string that is the address of the operand. %WHERE can be used only as the outermost expression in the LIST command.
Chapter 3. EQAOPTS options

The EQAOPTS load module allows customization of certain Debug Tool functions, which can apply to the site, a group, or a single user. This topic summarizes the purpose of each EQAOPTS option. See Debug Tool Customization Guide for information on how to create the EQAOPTS load module and a complete description of each option.

The follow diagram describes the syntax of this option:

```
<table>
<thead>
<tr>
<th>EQAOPT</th>
<th>BROWSE</th>
<th>RACF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>OFF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CACHENUM</th>
<th>cache_value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODEPAGE</td>
<td>nnnn</td>
</tr>
<tr>
<td>DEFAULTVIEW</td>
<td>STANDARD</td>
</tr>
<tr>
<td></td>
<td>NOMACGEN</td>
</tr>
<tr>
<td>DTCNFORCECUID</td>
<td>YES</td>
</tr>
<tr>
<td>DTCNFORCEIP</td>
<td>NO</td>
</tr>
<tr>
<td>DTCNFORCELOADMODID</td>
<td></td>
</tr>
<tr>
<td>DTCNFORCENETNAME</td>
<td></td>
</tr>
<tr>
<td>DTCNFORCETRANID</td>
<td></td>
</tr>
<tr>
<td>DTCNFORCETERMID</td>
<td></td>
</tr>
<tr>
<td>DTCNFORCETRANID</td>
<td></td>
</tr>
<tr>
<td>DTCNFORCEUSERID</td>
<td></td>
</tr>
<tr>
<td>EQAQPP</td>
<td>ON</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
</tr>
<tr>
<td>GPFDNS</td>
<td>file_name</td>
</tr>
<tr>
<td>MDBG</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>NO</td>
</tr>
<tr>
<td>NAMES</td>
<td>EXCLUDE</td>
</tr>
<tr>
<td></td>
<td>LOADMOD</td>
</tr>
<tr>
<td></td>
<td>pattern</td>
</tr>
<tr>
<td>INCLUDE</td>
<td>LOADMOD</td>
</tr>
<tr>
<td></td>
<td>name</td>
</tr>
<tr>
<td>NODISPLAY</td>
<td>DEFAULT</td>
</tr>
<tr>
<td>QUITDEBUG</td>
<td></td>
</tr>
<tr>
<td>SAVEBPOSN</td>
<td>userid-file-name</td>
</tr>
<tr>
<td>SAVESETDSN</td>
<td></td>
</tr>
<tr>
<td>SUBSYS</td>
<td>four-character-name</td>
</tr>
<tr>
<td>SVCSCREEN</td>
<td>SVCSCREEN_options</td>
</tr>
<tr>
<td>THREADTERMCOND</td>
<td>NOPROMPT</td>
</tr>
<tr>
<td></td>
<td>PROMPT</td>
</tr>
<tr>
<td>TIMACB</td>
<td>ACB-name</td>
</tr>
</tbody>
</table>

SVCSCREEN_options:

| SVCSCREEN | ON |
|           | OFF |
| CONFLICT   | OVERRIDE |
|           | NOOVERRIDE |
```
BROWSE

Specifies whether a user with sufficient RACF authority to the applicable browse mode RACF facility can start Debug Tool in browse mode.

Related tasks

“User enablement of browse mode” in Debug Tool Customization Guide

CACHENUM

Specifies the maximum number of program items to be held in an in-memory cache for a debug session. Increase this number to improve performance for applications which have many programs; however, a larger number also increases the storage usage by the debugged task.

Related tasks

“Overriding the default number of program elements held in cache” in Debug Tool Customization Guide

CODEPAGE

Indicates which code page to use so that NLS characters are properly communicated between Debug Tool and a remote debugger and properly displayed in full screen mode.

Related tasks

“Specifying a code page” in Debug Tool Customization Guide

DEFAULTVIEW

Provides a method of setting the initial value for the SET DEFAULT VIEW command.

Related tasks

“Setting the initial value for SET DEFAULT VIEW” in Debug Tool Customization Guide

DTCNFORCE

Controls DTCN behavior for the conditions described in Table 2. When you set a DTCNFORCE option to YES, DTCN forces users to specify the respective resource type. The default setting is NO.

Table 2. List of EQAXOPT options and its corresponding DTCN field

<table>
<thead>
<tr>
<th>EQAXOPT option</th>
<th>DTCN field name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTCNFORCECUID or DTCNFORCEPROGID</td>
<td>CU</td>
</tr>
<tr>
<td>DTCNFORCEIP</td>
<td>IP Name/Address</td>
</tr>
<tr>
<td>DTCNFORCELOADMOD</td>
<td>LoadMod</td>
</tr>
<tr>
<td>DTCNFORCENETNAME</td>
<td>NetName</td>
</tr>
<tr>
<td>DTCNFORCETERMID</td>
<td>Terminal Id</td>
</tr>
<tr>
<td>DTCNFORCETRANID</td>
<td>Transaction Id</td>
</tr>
</tbody>
</table>
Table 2. List of EQAXOPT options and its corresponding DTCN field  (continued)

<table>
<thead>
<tr>
<th>EQAXOPT option</th>
<th>DTCN field name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTCNFORCEUSERID</td>
<td>User Id</td>
</tr>
</tbody>
</table>

**Related tasks**

“Requiring users to specify resource types” in [Debug Tool Customization Guide](#).

---

**EQAQPP**

Indicates the presence of Q++ programs.

**Related tasks**

“Configuring for debugging Q++ programs” in [Debug Tool Customization Guide](#).

---

**GPFDSN**

Specifies the data set name for the global preferences file.

**Related tasks**

“Specifying global preferences” in [Debug Tool Customization Guide](#).

---

**MDBG**

For z/OS XL C/C++, Version 1 Release 11, if you have compiled with the DEBUG(FORMAT(DWARF)) compiler option, specifies whether Debug Tool obtains debug data from .mdbg or .dbg files.

**Related tasks**

“Specifying whether Debug Tool searches for .mdbg files” in [Debug Tool Customization Guide](#).

---

**NAMES**

Provides a method of entering NAMES commands that apply before the first load module and any of the compile units contained in that load module are processed.

**Related tasks**

“Supplying NAMES commands for the initial load module” in [Debug Tool Customization Guide](#).

---

**NODISPLAY**

Controls Debug Tool behavior when the terminal using full-screen mode using a dedicated terminal or the remote debugger are not available.

**Related tasks**

“Modifying Debug Tool behavior when requested user interface is not available” in [Debug Tool Customization Guide](#).

---

**SAVEBPDSN and SAVESETDSN**

Specifies the data set names to be used to save the breakpoints (SAVEBPDSN) and settings (SAVESETDSN). One qualifier in each of these data set names should be &&USERID, which represents the user ID of the current user.

**Related tasks**

“Modifying the name of the default data sets that store settings, breakpoints, and monitor values” in [Debug Tool Customization Guide](#).
**SUBSYS**

Provides a 1 to 4 character subsystem name. If an Enterprise PL/I or C/C++ source file is found to have a DSORG of DA or VSAM and this parameter is supplied, then this parameter is passed to SVC 99 (dynamic allocation) through the SUBSYS text unit when Debug Tool allocates the source file.

**Related tasks**

- "Specifying SUBSYS to access source code in a library system" in Debug Tool Customization Guide

---

**SVCSCREEN**

Controls Debug Tool’s enablement of SVC screening.

**Related tasks**

- "Setting the SVC screening option" in Debug Tool Customization Guide

---

**THREADTERMCOND**

Specifies whether Debug Tool should suppress the prompt it displays when the thread termination condition, FINISH condition, or CEE067 is raised by Language Environment.

**Related tasks**

- "Suppressing the prompt Debug Tool displays for FINISH, CEE066, or CEE067 conditions" in Debug Tool Customization Guide

---

**TIMACB**

Specifies an alternate Terminal Information Manager ACB name.

**Related tasks**

- "Running the Terminal Interface Manager on more than one LPAR on the same VTAM network" in Debug Tool Customization Guide
Notices

This information was developed for products and services offered in the U.S.A. IBM might not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
3-2-12, Roppongi, Minato-ku, Tokyo 106-8711

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with the local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement might not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.
Copyright license

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or functions of these programs.

Programming interface information

This book is intended to help you debug application programs. This publication documents intended Programming Interfaces that allow you to write programs to obtain the services of Debug Tool.

Trademarks and service marks

IBM, the IBM logo, and ibm.com® are 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” at www.ibm.com/legal/copytrade.shtml.

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

LINUX is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT®, and the Windows logo 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.

MasterCraft is a trademark of Tata Consultancy Services Ltd.
Readers’ Comments — We’d Like to Hear from You

Debug Tool for z/OS
Reference Summary
Version 10.1

Publication No. GC14-7256-00

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.
Submit your comments using one of these channels:
• Send your comments to the address on the reverse side of this form.
• Send your comments via e-mail to: COMMENTS@US.IBM.COM

If you would like a response from IBM, please fill in the following information:

Name

Address

Company or Organization

Phone No.

E-mail address