

IBM z/OS Debugger  
15.0.0

*Reference Summary*



**Note!**

Before using this information and the product it supports, be sure to read the general information under [“Notices” on page 77](#).

**Seventh Edition (November 2022)**

This edition applies to IBM® z/OS® Debugger, 15.0.6 (Program Number 5724-T07 with the PTF for PH46537), which supports the following compilers:

- Open Enterprise SDK for Go 1.17 (Program Number 5655-GOZ)
- z/OS XL C/C++ Version 2 (Program Number 5650-ZOS)
- C/C++ feature of z/OS Version 1 (Program Number 5694-A01)
- C/C++ feature of OS/390® (Program Number 5647-A01)
- C/C++ for MVS/ESA Version 3 (Program Number 5655-121)
- AD/Cycle C/370 Version 1 Release 2 (Program Number 5688-216)
- Enterprise COBOL for z/OS 6.1, 6.2, 6.3, and 6.4 (Program Number 5655-EC6)
- Enterprise COBOL for z/OS Version 5 (Program Number 5655-W32)
- Enterprise COBOL for z/OS Version 4 (Program Number 5655-S71)
- Enterprise COBOL for z/OS and OS/390 Version 3 (Program Number 5655-G53)
- COBOL for OS/390 & VM Version 2 (Program Number 5648-A25)
- COBOL for MVS™ & VM Version 1 Release 2 (Program Number 5688-197)
- COBOL/370 Version 1 Release 1 (Program Number 5688-197)
- VS COBOL II Version 1 Release 3 and Version 1 Release 4 (Program Numbers 5668-958, 5688-023) - with limitations
- OS/VS COBOL, Version 1 Release 2.4 (5740-CB1) - with limitations
- High Level Assembler for MVS & VM & VSE Version 1 Release 4, Version 1 Release 5, Version 1 Release 6 (Program Number 5696-234)
- Enterprise PL/I for z/OS 6.1 (Program Number 5655-PL6)
- Enterprise PL/I for z/OS Version 5 Release 1, Release 2, and Release 3 (Program Number 5655-PL5)
- Enterprise PL/I for z/OS Version 4 (Program Number 5655-W67)
- Enterprise PL/I for z/OS and OS/390 Version 3 (Program Number 5655-H31)
- VisualAge® PL/I for OS/390 Version 2 Release 2 (Program Number 5655-B22)
- PL/I for MVS & VM Version 1 Release 1 (Program Number 5688-235)
- OS PL/I Version 2 Release 1, Version 2 Release 2, Version 2 Release 3 (Program Numbers 5668-909, 5668-910) - with limitations

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

You can find out more about IBM z/OS Debugger by visiting the following IBM Web sites:

- IBM Debug for z/OS: <https://www.ibm.com/products/debug-for-zos>
- IBM Developer for z/OS: <https://www.ibm.com/products/developer-for-zos>
- IBM Z and Cloud Modernization Stack: <https://www.ibm.com/docs/z-modernization-stack>

© **Copyright International Business Machines Corporation 1992, 2022.**

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

---

# Contents

<b>About this document.....</b>	<b>ix</b>
Who might use this document.....	ix
Accessing z/OS licensed documents on the Internet.....	ix
How to read syntax diagrams.....	x
Symbols.....	x
Syntax items.....	x
Syntax examples.....	xi
How to provide your comments.....	xii
 <b>Summary of changes.....</b>	 <b>xiii</b>
 <b>Overview of IBM z/OS Debugger.....</b>	 <b>xxi</b>
 <b>Chapter 1. z/OS Debugger commands.....</b>	 <b>1</b>
? command.....	1
ALLOCATE command.....	1
ANALYZE command (PL/I).....	1
Assignment command (assembler and disassembly).....	1
Assignment command (LangX COBOL).....	1
Assignment command (PL/I).....	2
AT ALLOCATE (PL/I).....	2
AT APPEARANCE.....	2
AT CALL.....	2
AT CHANGE.....	2
AT CHANGE (remote).....	3
AT CURSOR (full-screen mode).....	3
AT DATE (COBOL).....	3
AT DELETE.....	3
AT ENTRY.....	4
AT ENTRY (remote).....	4
AT EXIT.....	4
AT GLOBAL.....	4
AT GLOBAL LABEL (remote).....	5
AT LABEL.....	5
AT LABEL (remote).....	5
AT LINE.....	6
AT LOAD.....	6
AT LOAD (remote).....	6
AT OCCURRENCE.....	6
AT OFFSET (disassembly).....	6
AT PATH.....	7
AT Prefix (full-screen mode).....	7
AT STATEMENT.....	7
AT STATEMENT (remote).....	7
AT TERMINATION.....	7
BEGIN command (programming language neutral).....	7
block command (C and C++).....	8
break command (C and C++).....	8
CALL %CEBR.....	8
CALL %CECI.....	8

CALL %DUMP.....	8
CALL %FA.....	8
CALL %FM.....	8
CALL %HOGAN.....	9
CALL %VER.....	9
CALL entry_name (COBOL).....	9
CALL procedure.....	9
CC command.....	9
CHKSTGV.....	9
CLEAR command.....	10
CLEAR AT (remote).....	11
CLEAR prefix (full-screen mode).....	12
COMMENT command.....	12
COMPUTE command (COBOL).....	12
CURSOR command (full-screen mode).....	12
Declarations (assembler, disassembly, and LangX COBOL).....	12
Declarations (C and C++).....	13
Declarations (COBOL).....	15
DECLARE command (PL/I).....	15
DESCRIBE command.....	16
DISABLE command.....	17
DISABLE prefix (full-screen mode).....	18
DO command (assembler, disassembly, LangX COBOL, and COBOL).....	18
DO command (PL/I).....	18
do/while command (C and C++).....	19
ENABLE command.....	19
ENABLE prefix (full-screen mode).....	20
EVALUATE command (COBOL).....	20
Expression command (C and C++).....	21
FIND command.....	21
FINDBP command.....	21
for command (C and C++).....	21
FREE command.....	21
GO command.....	21
GOTO command.....	22
GOTO LABEL command.....	22
%IF command (programming language neutral).....	22
IF command (assembler, disassembly, and LangX COBOL).....	22
if command (C and C++).....	22
IF command (COBOL).....	22
IF command (PL/I).....	23
IMMEDIATE command (full-screen mode).....	23
INPUT command (C and C++ and COBOL).....	23
JUMPTO command.....	23
JUMPTO LABEL command.....	23
LIST (blank).....	23
LIST AT.....	24
LIST AT (remote).....	24
LIST CALLS.....	25
LIST CC command.....	25
LIST CONTAINER.....	25
LIST CURSOR (full-screen mode).....	25
LIST DTCN or CADP.....	25
LIST expression.....	26
L expression prefix.....	26
LIST FREQUENCY.....	26
LIST LAST.....	27
LIST LDD.....	27

LIST LINE NUMBERS.....	27
LIST LINES.....	27
LIST MONITOR.....	27
LIST NAMES.....	27
LIST NAMES LABELS (remote).....	28
LIST ON (PL/I).....	28
LIST PROCEDURES.....	28
LIST REGISTERS.....	28
LIST STATEMENT NUMBERS.....	29
LIST STATEMENTS.....	29
LIST STORAGE.....	29
LIST TRACE LOAD command.....	29
LOAD command.....	29
LOADDEBUGDATA (LDD).....	30
MEMORY.....	30
M prefix command.....	30
MONITOR command.....	31
MOVE command (COBOL).....	31
NAMES DISPLAY command.....	31
NAMES EXCLUDE command.....	32
NAMES INCLUDE command.....	32
Null command.....	32
ON command (PL/I).....	32
PANEL command (full-screen mode).....	33
PERFORM command (COBOL).....	33
PLAYBACK BACKWARD command.....	34
PLAYBACK DISABLE command.....	34
PLAYBACK ENABLE command.....	34
PLAYBACK FORWARD command.....	34
PLAYBACK START command.....	35
PLAYBACK STOP command.....	35
POPUP command.....	35
POSITION command.....	35
Prefix commands (full-screen mode).....	35
PROCEDURE command.....	36
QUALIFY RESET.....	36
QUERY command.....	36
QUERY prefix (full-screen mode).....	39
QUIT command.....	39
QQUIT command.....	39
RESTORE command.....	39
RETRIEVE command (full-screen mode).....	39
RUN command.....	39
RUNTO command.....	39
RUNTO prefix command (full-screen mode).....	40
SCROLL command (full-screen mode).....	40
SELECT command (PL/I).....	40
SET ASSEMBLER ON/OFF.....	40
SET ASSEMBLER STEPOVER.....	41
SET AUTOMONITOR.....	41
SET CHANGE.....	41
SET COLOR (full-screen and line mode).....	41
SET COUNTRY.....	42
SET DBCS.....	42
SET DEFAULT DBG.....	42
SET DEFAULT LISTINGS.....	43
SET DEFAULT MDBG.....	43
SET DEFAULT SCROLL (full-screen mode).....	43

SET DEFAULT VIEW.....	43
SET DEFAULT WINDOW (full-screen mode).....	44
SET DISASSEMBLY.....	44
SET DYNDEBUG.....	44
SET ECHO.....	44
SET EQUATE.....	44
SET EXECUTE.....	44
SET EXPLICITDEBUG.....	45
SET FIND BOUNDS.....	45
SET FREQUENCY.....	45
SET HISTORY.....	45
SET IGNORELINK.....	45
SET INTERCEPT (C and C++).....	45
SET INTERCEPT (COBOL, full-screen mode, line mode, batch mode).....	46
SET INTERCEPT (COBOL, remote debug mode).....	46
SET KEYS (full-screen and line mode).....	46
SET LDD.....	46
SET LIST BY SUBSCRIPT command (COBOL).....	46
SET LIST BY SUBSCRIPT command (Enterprise PL/I, full-screen mode only).....	46
SET LIST TABULAR.....	47
SET LOG.....	47
SET LOG NUMBERS (full-screen and line mode).....	47
SET LONGCUNAME (C, C++, and PL/I).....	47
SET MDBG (C, C++).....	47
SET MONITOR (full-screen and line mode).....	47
SET MSGID.....	48
SET NATIONAL LANGUAGE.....	48
SET PACE.....	48
SET PFKEY.....	48
SET POPUP.....	48
SET PROGRAMMING LANGUAGE.....	48
SET PROMPT (full-screen and line mode).....	49
SET QUALIFY.....	49
SET REFRESH (full-screen mode).....	49
SET RESTORE.....	49
SET REWRITE (full-screen mode).....	50
SET REWRITE (remote debug mode).....	50
SET SAVE.....	50
SET SCREEN (full-screen and line mode).....	50
SET SCROLL DISPLAY (full-screen mode).....	51
SET SEQUENCE (PL/I).....	51
SET SOURCE.....	51
SET SUFFIX (full-screen mode).....	51
SET TEST.....	51
SET WARNING (C, C++, and PL/I).....	51
SET command (COBOL).....	52
SHOW prefix command (full-screen mode).....	52
STEP command.....	52
STORAGE command.....	52
switch command (C and C++).....	52
SYSTEM command.....	53
TRACE command.....	53
TRIGGER command.....	53
TSO command (z/OS).....	55
USE command.....	55
while command (C and C++).....	55
WINDOW CLOSE.....	55
WINDOW OPEN.....	55

WINDOW SIZE.....	55
WINDOW SWAP.....	56
WINDOW ZOOM.....	56
<b>Chapter 2. z/OS Debugger built-in functions.....</b>	<b>57</b>
%CHAR (assembler, disassembly, and LangX COBOL).....	57
%DEC (assembler, disassembly, and LangX COBOL).....	57
%GENERATION (PL/I).....	57
%HEX.....	57
%INSTANCES (C, C++, and PL/I).....	57
%RECURSION (C, C++, and PL/I).....	57
%WHERE (assembler, disassembly, and LangX COBOL).....	57
<b>Chapter 3. EQAOPTS commands (optional).....</b>	<b>59</b>
ALTDISP.....	61
BROWSE.....	61
CACHENUM.....	61
CCOUTPUTDSN.....	62
CCOUTPUTDSNALLOC.....	62
CCPROGSELECTDSN.....	62
CEEREACTAFTERQDBG.....	62
CODEPAGE.....	62
COMMANDSDSN.....	62
DEFAULTVIEW.....	62
DISABLERLIM.....	62
DLAYDBG.....	62
DLAYDBGEND.....	62
DLAYDBGDSN.....	63
DLAYDBGTRC.....	63
DLAYDBGXRF.....	63
DTCNDELETEDEADPROF.....	63
DTCNFORCExxxx.....	63
DYNDEBUG.....	63
EQAQPP.....	63
EXPLICITDEBUG.....	63
GPFDSN.....	64
HOSTPORTS.....	64
IGNOREODOLIMIT.....	64
IMISOORIGPSB.....	64
LOGDSN.....	64
LOGDSNALLOC.....	64
MAXTRANUSER.....	64
MDBG.....	64
MULTIPROCESS.....	64
NAMES.....	64
NODISPLAY.....	65
PREFERENCESDSN.....	65
SAVEBPDSN and SAVESETDSN.....	65
SAVEBPDSNALLOC and SAVESETDSNALLOC.....	65
SESSIONTIMEOUT.....	65
STARTSTOPMSG.....	65
STARTSTOPMSGDSN.....	65
SUBSYS.....	65
SVCSCREEN.....	65
TCPIPDATAIDSN.....	65
THREADTERMCOND.....	65
TIMACB.....	66

END.....	66
<b>Chapter 4. z/OS Debugger variables.....</b>	<b>67</b>
<b>Chapter 5. Reference card: Frequently used z/OS Debugger commands.....</b>	<b>69</b>
<b>Chapter 6. Reference card: Frequently used z/OS Debugger commands while debugging assembler programs.....</b>	<b>73</b>
<b>Notices.....</b>	<b>77</b>
Copyright license.....	77
Privacy policy considerations.....	78
Programming interface information.....	78
Trademarks and service marks.....	78



## About this document

---

z/OS Debugger 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. z/OS Debugger 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 commands provided by z/OS Debugger. Each topic contains the name of the command, built-in function, or EQAOPTS command and then a syntax diagram. For more information about each command or built-in function, refer to the *IBM z/OS Debugger Reference and Messages*. For more information about each EQAOPTS command, see the *IBM z/OS Debugger Customization Guide* or *IBM z/OS Debugger Reference and Messages*.

## Who might use this document

---

This document is intended for programmers using z/OS Debugger 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.

z/OS Debugger 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 the Terminal Interface Manager 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/resource link>

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, (GI10-8928), that includes this key code.

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

<http://www.ibm.com/servers/resource link>

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.

## How to read syntax diagrams

---

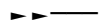
This section describes how to read syntax diagrams. It defines syntax diagram symbols, items that may be contained within the diagrams (keywords, variables, delimiters, operators, fragment references, operands) and provides syntax examples that contain these items.

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:

Symbol	Definition
--------	------------



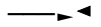
Indicates the beginning of the syntax diagram.



Indicates that the syntax diagram is continued to the next line.



Indicates that the syntax is continued from the previous line.



Indicates the end of the syntax diagram.

### 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.

Item type	Definition
-----------	------------

<b>Required</b>	Required items are displayed on the main path of the horizontal line.
-----------------	---

<b>Optional</b>	Optional items are displayed below the main path of the horizontal line.
-----------------	--

<b>Default</b>	Default items are displayed above the main path of the horizontal line.
----------------	---

# Syntax examples

The following table provides syntax examples.

Table 1. Syntax examples

Item	Syntax example
Required item. Required items appear on the main path of the horizontal line. You must specify these items.	►► KEYWORD — required_item ◄◄
Required choice. 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.	►► KEYWORD { required_choice1 required_choice2 } ◄◄
Optional item. Optional items appear below the main path of the horizontal line.	►► KEYWORD { optional_item } ◄◄
Optional choice. 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.	►► KEYWORD { optional_choice1 optional_choice2 } ◄◄
Default. 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.	►► KEYWORD { default_choice1 optional_choice2 optional_choice3 } ◄◄
Variable. Variables appear in lowercase italics. They represent names or values.	►► KEYWORD — <i>variable</i> ◄◄
Repeatable item. An arrow returning to the left above the main path of the horizontal line indicates an item that can be repeated. A character within the arrow means you must separate repeated items with that character. 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.	►► KEYWORD { repeatable_item } ◄◄ ►► KEYWORD { repeatable_item , } ◄◄ ►► KEYWORD { repeatable_item } ◄◄

Table 1. Syntax examples (continued)

Item	Syntax example
<p>Fragment.</p> <p>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.</p>	<p>The diagram illustrates the fragment symbol and a complex choice structure. The top part shows the fragment symbol: a double arrow pointing right, followed by the text 'KEYWORD', then a box containing the word 'fragment', and finally another double arrow pointing right. Below this, the word 'fragment' is written in bold. The main diagram shows a horizontal line starting with a double arrow pointing right. This line branches into two paths. The upper path contains the text ', — required_choice1' and ends with a double arrow pointing right. The lower path contains the text ', — required_choice2' and leads to a vertical bracket. To the right of this bracket are two options: ', — default_choice' and ', — optional_choice'. Both options are connected to the main horizontal line by vertical lines, and the entire structure ends with a double arrow pointing right.</p>

## How to provide 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 z/OS Debugger documentation, you can leave a comment in [IBM Documentation](#):

- IBM Developer for z/OS and IBM Developer for z/OS Enterprise Edition: <https://www.ibm.com/docs/en/developer-for-zos>
- IBM Debug for z/OS: <https://www.ibm.com/docs/debug-for-zos>
- IBM Z and Cloud Modernization Stack: <https://www.ibm.com/docs/z-modernization-stack>

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

---

## 15.0.6

### Debug Profile Service

- On the **IBM z/OS Debugger Preferences** page, you can specify to ignore the SSL certificate errors when the Debug Profile Service that you want to connect to does not have a valid SSL certificate. For more information, see the "Setting debug preferences" topic in [IBM Documentation](#).

## 15.0.5

### z16, Compilers and CICS

- Support is added for the new IBM z16 hardware.
- Support is added for CICS Transaction Server for z/OS 6.1.
- Support is added for the following compiler versions in both 31-bit mode and 64-bit mode:
  - Enterprise COBOL for z/OS 6.4
  - Enterprise PL/I for z/OS 6.1

### User-defined functions

- User-defined functions are supported for programs compiled with Enterprise COBOL for z/OS 6.4.

### 64-bit support

- Debug Tool compatibility mode now supports delay debug.
- The following EQAOPTS commands are now supported:
  - DLAYDBG
  - DLAYDBGDSN
  - DLAYDBGTRC

For the remaining limitations, see "Limitations of 64-bit support in Debug Tool compatibility mode" in *IBM z/OS Debugger User's Guide*.

### Interoperability

- Interoperability is now supported between 31-bit and 64-bit COBOL programs. Use delay debug mode to improve efficiency.
- You can now debug 31-bit COBOL programs called from 64-bit Java programs. Delay debug mode is required when you debug this type of application.

For more information, see "Using delay debug mode to delay starting of a debug session" in *IBM z/OS Debugger User's Guide*.

### Automatic Binary Optimizer for z/OS

- Support is added for Automatic Binary Optimizer for z/OS 2.2.
- In Debug Tool compatibility mode, you can now debug programs compiled with Enterprise COBOL for z/OS Version 5 or later that have been optimized by Automatic Binary Optimizer for z/OS 2.2. The optimized programs can be debugged in the same manner as those that are not optimized.

### Code Coverage

- You can now filter code coverage results for z/OS batch applications using existing JCL and z/OS UNIX applications by specifying a filter list file in the launch configuration. For more information, see "Filtering code coverage results during collection" in [IBM Documentation](#).
- The module exclude list is deprecated and replaced by the filter list file.

## Debug profile

- As a system administrator of Debug Profile Service, you can now use z/OS Debugger Profile Management to identify and delete CICS (DTCN) profiles that might interfere with z/OS system resources. For more information, see "Managing profiles with z/OS Debugger Profile Management" in *IBM z/OS Debugger Customization Guide*.

## Host configuration

- When not run as a started task, Debug Manager now queries your security product for explicit permission to start. For more information, see "Starting Debug Manager as a user job" in *IBM z/OS Debugger Customization Guide*.

## 15.0.4

### IBM Z® Open Debug

- You can now connect to a debug session and start debugging when you list debug sessions that are parked on a z/OS machine.
- With the Wazi for VS Code IDE, you can now use a single action to activate a debug profile, launch and debug an application.

## Code Coverage

- You can now merge and export code coverage results on z/OS from command line into a single file of various formats with the `ccexport.sh` command. For more information, see "Merging and exporting code coverage results from z/OS" in [IBM Documentation](#).
- Code Coverage Service can now be started as part of Remote Debug Service. For more information, see "Generating code coverage in headless mode using Remote Debug Service" in [IBM Documentation](#).
- When started via the headless code coverage collector, Code Coverage Service now supports secured connections, and requires authentication.
- In the **Code Coverage Results** view, you can now add a secured Code Coverage Service result location (https). You can add and clear untrusted certificates in the CCS keystore file. For more information, see "Working with result locations" in [IBM Documentation](#).
- The code coverage output location specified in the `o, output` parameter is ignored in the startup key unless you specify `-a, allowoutputlocation=TRUE` in the command line when you start the code coverage collector and use headless code coverage. For more information, see "Specifying code coverage options in the startup key" and "Starting and stopping the headless code coverage daemon" in [IBM Documentation](#).
- The Code Coverage API documentation is updated from version 10.1.2 to 11.0.0.

### IBM Open Enterprise SDK for Go

- In Debug Tool compatibility mode, you can now debug Go programs compiled with IBM Open Enterprise SDK for Go 1.17. For more information, see "Debugging programs compiled with IBM Open Enterprise SDK for Go" in *IBM z/OS Debugger User's Guide*.

## Decimal point

- In Debug Tool compatibility mode, if the `DECIMAL-POINT IS COMMA` clause is specified in a COBOL program compiled with Enterprise COBOL for z/OS Version 6 Release 3 (UI78163) or later, the debugger displays decimals as commas in the **Variables** and **Monitors** views in Eclipse IDEs, and expressions accept commas, in addition to periods, as decimal points.

## Host configuration

- You can now configure the `eqahcc.enc` file to start Code Coverage Service as part of Remote Debug Service. For more information, see the "Customizing with the sample job EQARMTSU" topic in *IBM z/OS Debugger Customization Guide*.

- With Debug Manager, you can leverage Dynamic Virtual IP Addressing (DVIPA) available in IBM Explorer for z/OS to concurrently run identical setups on different systems in your sysplex, and have TCP/IP, optionally with the help of WLM, distribute the client connections among these systems. Ensure that each Debug Manager has a unique external port per system and the port is explicitly defined in TCP/IP definitions. For more information, see the "Distributed Dynamic VIPA" section in *IBM z/OS Debugger Customization Guide*.
- MVS data set `userid.EQATIOUT` is no longer needed when you install and configure the IMS transaction isolation extension for Eclipse IDE users.

## 15.0.3

### z/OS 2.5

- Support is added for z/OS 2.5.

### IBM Open Enterprise SDK for Go

- In Debug Tool compatibility mode, you can now debug Go programs compiled with IBM Open Enterprise SDK for Go 1.16. For more information, see "Debugging programs compiled with IBM Open Enterprise SDK for Go" in *IBM z/OS Debugger User's Guide*.

### 64-bit support

- Debug Tool compatibility mode now supports playback for 64-bit COBOL programs. For the remaining limitations, see "Limitations of 64-bit support in Debug Tool compatibility mode" in *IBM z/OS Debugger User's Guide*.

### Source Level Code Coverage for COBOL

- When you start a code coverage session with the Eclipse IDE or headless code coverage, you can now choose to use the source listing. Source level code coverage offers direct mapping between code coverage entries and the program source, to exclude the need to post process the code coverage data. Source level code coverage improves integration with tools like ZUnit and SonarQube as part of an automated pipeline. For more information, see "How does z/OS Debugger locate COBOL source during code coverage" in *IBM z/OS Debugger User's Guide*.

### Code Coverage Service API

- Code Coverage Service (CCS) RESTful API is now available to enable custom extensions. For more information, see "Code Coverage Service RESTful API Documentation" in [IBM Documentation](#).

### z/OS Debugger Profiles view

- Remote IMS Application with Isolation launch configurations have been replaced by the IMS Isolation debug profiles. All existing IMS launch configurations are automatically migrated to the **z/OS Debugger Profiles view**. You can create and activate IMS Isolation profiles in the view to debug and run code coverage for IMS transactions in private regions.

To use this function, ensure that the system programmer installed and configured the IMS transaction isolation extension for the ADFz Common Components server. If you want to configure the region name for the private region, ask the system programmer to update z/OS Debugger to 15.0.3 or later, with the PTF for APAR PH41774 applied.

**Note:** IMS Isolation profiles are only available in IBM Developer for z/OS Enterprise Edition.

- You can now view the Remote System Explorer z/OS connection status in the view. A new option **Refresh z/OS Connections** is provided in the view toolbar to establish all z/OS connections and synchronize the profiles.
- You can now duplicate the content from an existing debug profile to create a new one efficiently.
- Generic profiles might trigger z/OS Debugger unexpectedly and consume unnecessary resources. When you activate a generic profile, warnings are now displayed. You can choose to hide the warnings.

- In the Debug Profile Editor, you can now save a debug profile without activating it, and leave it for future use.

For more information, see "Managing debug profiles with the z/OS Debugger Profiles view" in [IBM Documentation](#).

### Debug Profile Service

- You now only need to expose one port to use Debug Profile Service. A new configuration switch is added to `eqaprof.env` to select whether to use secure HTTP protocol. For more information, see "Customizing with the sample job EQAPRFSU" in *IBM z/OS Debugger Customization Guide*.

### z/OS Debugger commands

- The following commands are now supported in Debug Tool compatibility mode for remote debugging:
  - STEP
  - GO
  - RUNTO
  - JUMPTO
  - COMMENT

### Dark theme

- Dark theme is now supported for remote debugger in the Eclipse IDE.

### Message EQA9924U

- Message ICH408I is now issued in the console with EQA9924U to provide you with more information to address the issue.

## 15.0.2

### IBM Z Open Debug

- You can now debug High Level Assembler (HLASM) z/OS programs with IBM Z Open Debug.
- For Wazi for Dev Spaces, the log files are now in `/projects/.debug/logs`.

### Code Coverage

- In the **Code Coverage Results** view, you can now export code coverage results in Cobertura format. For more information, see "Exporting code coverage results in Cobertura format" in [IBM Documentation](#).
- You can now specify parameters in the startup key to generate code coverage results in Cobertura and SonarQube formats. In addition, short parameters values `-e, exportertype=SQ|PDF|COB` are added for you to use both in the startup key and in the headless code coverage daemon. For more information, see "Specifying code coverage options in the startup key" and "Starting and stopping the headless code coverage daemon" in [IBM Documentation](#).
- When you view code coverage results in an editor, you can now see a code coverage summary of the included files for PL/I source files with `%INCLUDE` statements. For more information, see "Viewing code coverage results in an editor" in [IBM Documentation](#).

### z/OS Batch Applications launches

- In the **Remote Systems** or **z/OS Projects** view, or when you are editing the JCL source in the editor, after you choose **Debug As** or **Code Coverage As** from the menu, the following options are available:
  - **z/OS Batch Application**: Launch a debug or code coverage session without a debug profile.
  - **z/OS Batch Application with a debug profile**: Launch a debug or code coverage session with a debug profile.



- **z/OS Batch Application ...**: Create a launch configuration to launch a debug or code coverage session.

For more information, see "Launching a debug session for z/OS batch applications using existing JCL" in *IBM Documentation*.

### IBM z/OS Debugger JCL Wizard

- The **Program/Procedure Selection List** panel is updated to include procedures, in addition to programs. Selecting a procedure will provide a panel to enter the procedure step override for the DD statements generated. The After (A) and Before (B) line commands are no longer required.
- You can now select **SVC screening** to enable SVC screening for batch non-Language Environment programs.
- You can now select **Intercept on** to show COBOL DISPLAY statements on the IBM z/OS Debugger log or Debug Console in the Eclipse IDE.
- Error messages are improved.
- The END command (**PF3**) in the **Program/Procedure Selection List** panel is modified to cancel the request. Previously, selecting **PF3** would not exit the panel.
- When the process is completed, the cursor is now placed on the command line.
- The **z/OS Debugger LDD Generation for Non-LE Programs** panel is now populated with the initial program name and subprograms selected in the **Request AT ENTRY Sub-Program Breakpoints** panel. Program names provided in this panel can be modified.
- After you select **Code Coverage** from the parameters selection panel, the EQAXOPT lines are generated to specify CCPROGSELECTDSN, CCOUTPUTDSN and CCOUTPUTDSNALLOC, if the CODE\_COVERAGE\_SETUP value is configured to YES in the EQAJCL REXX procedure.
- Previously the wizard would verify the program source members identified in the **z/OS Debugger LDD Generation for Non-LE Programs** panel with each library identified by the **z/OS Debugger Debug Libraries** panel to verify that the members are present. This function is removed because z/OS Debugger now flags any such members in the IBM z/OS Debugger log or Debug Console in the Eclipse IDE when the LDD command is entered.

For more information, see "IBM z/OS Debugger JCL Wizard" in the *IBM z/OS Debugger User's Guide*

### AT LABEL \* command

- For Enterprise COBOL for z/OS Version 5 and later, AT LABEL \* now highlights the labels similar to statement breakpoints.
- You can now use PF6 or AT LINE to remove or add a single global label hook if AT LABEL \* was issued. To disable this functionality, use DISABLE AT LABEL \*.
- If you issue AT LABEL \* again or use ENABLE AT LABEL \*, the global label hooks are reset. The hooks that you removed are added back.

For more information, see "AT LABEL command" in *IBM z/OS Debugger Reference and Messages*.

## 15.0.1

### 64-bit support

- Debug Tool compatibility mode now supports the following features:
  - Code coverage
  - Source entry breakpoints
  - CEETEST

For the remaining limitations, see the "Limitations of 64-bit support in Debug Tool compatibility mode" topic in *IBM z/OS Debugger User's Guide*.

## Code Coverage

- With Concurrent Debug and Code Coverage, you can run code coverage collection in parallel with the active debug session in the Eclipse IDE. The code coverage data is collected during the debug run, and code coverage annotations are displayed and updated in the debug editor. Concurrent Debug and Code Coverage requires z/OS Debugger 15.0.1 or later. For more information, see the "Generating code coverage in a remote debug session" topic in [IBM Documentation](#).
- Headless code coverage report can now be exported with a Cobertura exporter. For more information, see the "Starting and stopping the headless code coverage daemon" topic in [IBM Documentation](#).
- Headless code coverage collector now supports filtering of module, compiler units, and files. For more information, see the "Filtering code coverage results" topic in [IBM Documentation](#).

## Debug Profile Editor

- In the Debug Profile Editor of the Eclipse IDE, new key bindings are available to show the error tooltip and the overall error summary. For more information, see the "Debug profile key bindings" topic in [IBM Documentation](#).

## Debug Profile Service

- As an alternative of a keystore file, you can now use a RACF managed key ring to enable secure communication with Debug Profile Service. For more information, see the "Enabling secure communication with a RACF managed key ring" section in *IBM z/OS Debugger Customization Guide*.
- A new optional HOST attribute is added to the CICS region configuration. For more information, see the instructions in the `/etc/debug/dtcn.ports` sample configuration file.
- The Debug Profile Service API now provides more detailed diagnostic messages when authentication fails.

## IBM Z Open Debug

- Log files can now be found in the user's home directory.

## CICS trace entries

- A new parameter, DNT, is added to the CICS startup parameter INITPARM to support disabling generation of z/OS Debugger trace entries. For more information, see the "Adding support for debugging under CICS" topic in *IBM z/OS Debugger Customization Guide*.

## 15.0.0

### 64-bit support

- Debug Tool compatibility mode now supports the following compiler features:
  - The 64-bit COBOL feature of z/OS for COBOL V6.3 and later
  - The 64-bit C/C++ feature of z/OS

For the limitations, see the "Limitations of 64-bit support in Debug Tool compatibility mode" topic in *IBM z/OS Debugger User's Guide*.

The PTFs for z/OS Language Environment APARs PH26071 and PH28997 are required for this support.

### IBM Z Open Debug

- IBM Z Open Debug is now also available with the Wazi for Dev Spaces IDE, in addition to the Wazi for VS Code IDE. Both IDEs are offered in IBM Wazi Developer for Red Hat CodeReady Workspaces and IBM Developer for z/OS Enterprise Edition. For a comparison of features provided in different products and IDEs, see [Overview of IBM z/OS Debugger](#).
- You can now specify `TEST( , , RDS:*)` for the TEST runtime option to start a debug session using Remote Debug Service for Wazi for VS Code or Wazi for Dev Spaces.

## Code Coverage

- Headless code coverage for z/OS is now included with IBM Debug for z/OS. Use the headless code coverage collector to generate code coverage results of tests that are run as part of your DEVOPS pipeline. For more information, see the "Generating code coverage in headless mode using a daemon" section in [IBM Documentation](#).
- Single letter parameters are now supported in the headless code coverage collector command line and in EQA\_STARTUP\_KEY when you use JCL. For more information, see topics "Starting and stopping the headless code coverage daemon" and "Specifying code coverage options in the startup key" in [IBM Knowledge Center](#).
- Support is added for PL/I programs compiled with LISTVIEW(SOURCE) to generate code coverage results for main program and all %INCLUDE files. For more information, see the "Supported compilers and options for code coverage in Debug Tool compatibility mode" topic in [IBM Knowledge Center](#).
- The **Code Coverage Results** view of the Eclipse IDE now supports CCS result locations. You can add a CCS result location which collects and retrieves code coverage data by using RESTful API, and interact with the results under the CCS result location in the same way as locally stored results. CCS result locations require Headless Code Coverage 15.0.0 or later. For more information, see the "Viewing code coverage results in the Code Coverage Results view" topic in [IBM Knowledge Center](#).
- You can now also use Remote Debug Service to collect code coverage results similar to the headless code coverage collector for IBM Wazi Developer for Red Hat CodeReady Workspaces or IBM Developer for z/OS Enterprise Edition. For more information, see the "Generating code coverage in headless mode using Remote Debug Service" topic in [IBM Knowledge Center](#).

## Debug Profile Editor

- In the Debug Profile Editor of the Eclipse IDE, you can now use the quick outline to navigate to a field. For more information, see the "Quick outline for the Debug Profile Editor" topic in [IBM Knowledge Center](#).

## z/OS XL C/C++

- Support is added for DEBUG(NOFIELD). For more information, see topics "Choosing DEBUG compiler suboptions for C programs" and "Choosing DEBUG compiler suboptions for C++ programs" in *IBM z/OS Debugger User's Guide*.

## Debug Tool Plug-ins

- The following Debug Tool plug-ins of the Eclipse IDE are deprecated and will be removed in the next release:
  - DTCN Profile Manager plug-in
  - DTSP Profile Manager plug-in
  - Instrument JCL for Debug Tool Debugging plug-in
  - Debug Tool Code Coverage plug-in
  - Load Module Analyzer plug-in

You can use the **z/OS Debugger Profiles** view to create and manage debug profiles, z/OS batch applications launches to dynamically instrument and submit JCL, and the **Code Coverage Results** view to work with compiled code coverage results. For more information, see the following topics in [IBM Documentation](#): Managing debug profiles with the z/OS Debugger Profiles view, Launching a debug session for z/OS batch applications using existing JCL, and Viewing code coverage results in the Code Coverage Results view.

## Load Module Analyzer

- The Load Module Analyzer is deprecated and will be removed in a future version.

## Host configuration

- Remote Debug Service can now be configured to collect headless code coverage. For more information, see the "Adding support for Remote Debug Service" section in *IBM z/OS Debugger Customization Guide*.
- The record size for the DTCN VSAM file is increased to 3000 bytes. To use the DTCN VSAM repository with z/OS Debugger 15.0, create a new file using the SEQASAMP (EQAWCRVS) sample JCL. You can also convert your existing VSAM file to the new record size and format using the EQADPCNV utility. For more information, see the "Migrating a debug profiles VSAM file from an earlier release" topic in *IBM z/OS Debugger Customization Guide*.
- The IMS Transaction Isolation Facility is enhanced to utilize type 2 IMS commands for retrieving information on transactions, in cases where the type 1 commands that are normally used are disallowed. For more information, see the "Scenario F: Enabling the Transaction Isolation Facility" topic in *IBM z/OS Debugger Customization Guide*.

# Overview of IBM z/OS Debugger

---

IBM z/OS Debugger is the next iteration of IBM debug technology on IBM Z and consolidates the IBM Integrated Debugger and IBM Debug Tool engines into one unified technology. IBM z/OS Debugger is progressing towards one remote debug mode based on Debug Tool compatibility mode. In support of this direction, Debug Tool compatibility mode, when available in the user interface, is selected by default for V14.1.2 or later.

IBM z/OS Debugger is a host component that supports various debug interfaces, like the Eclipse and Visual Studio Code IDEs. z/OS Debugger and the supported debug interfaces are provided with the following products:

## **IBM Developer for z/OS Enterprise Edition**

This product is included in [IBM Application Delivery Foundation for z/OS](#). IBM Developer for z/OS Enterprise Edition provides all the debug features.

IBM Developer for z/OS Enterprise Edition currently provides debug functions in the following IDEs:

- IBM Developer for z/OS Eclipse
- Wazi for Dev Spaces, through IBM Z Open Debug
- Wazi for VS Code, through IBM Z Open Debug

See [Table 3 on page xxiii](#) for the debug features supported in different IDEs.

## **IBM Developer for z/OS**

IBM Developer for z/OS is a subset of IBM Developer for z/OS Enterprise Edition. IBM Developer for z/OS, previously known as IBM Developer for z Systems or IBM Rational® Developer for z Systems®, is an Eclipse-based integrated development environment for creating and maintaining z/OS applications efficiently.

IBM Developer for z/OS includes all enhancements in IBM Developer for z/OS Enterprise Edition except for the debug features noted in [Table 2 on page xxii](#).

## **IBM Debug for z/OS**

IBM Debug for z/OS is a subset of IBM Developer for z/OS Enterprise Edition. IBM Debug for z/OS focuses on debugging solutions for z/OS application developers. See [Table 2 on page xxii](#) for the debug features supported.

IBM Debug for z/OS does not provide advanced developer features that are available in IBM Developer for z/OS Enterprise Edition.

For information about how to install the IBM Debug for z/OS Eclipse IDE, see [Installing the IBM Debug for z/OS Eclipse IDE](#).

## **IBM Z and Cloud Modernization Stack**

IBM Z and Cloud Modernization Stack brings together component capabilities from IBM Z into an integrated platform that is optimized for Red Hat OpenShift Container Platform. With this solution, you can analyze the impact of application changes on z/OS, create and deploy APIs for z/OS applications, work on z/OS applications with cloud native tools, and standardize ID automation for z/OS. Starting from 2.0, Wazi Code is delivered in IBM Z and Cloud Modernization Stack. Wazi Code 1.x is still available in IBM Wazi Developer for Red Hat CodeReady Workspaces.

The debug functions are available in the IDEs provided with Wazi Code:

- Wazi for Dev Spaces, through IBM Z Open Debug
- Wazi for VS Code, through IBM Z Open Debug
- Wazi for Eclipse

See [Table 2 on page xxii](#) and [Table 3 on page xxiii](#) for the debug features supported in the product and different IDEs.

Table 2 on page xxii maps out the features that differ in products. Not all the available features are listed. To find the features available in different remote IDEs, see Table 3 on page xxiii.

Table 2. Debug feature comparison				
	IBM Debug for z/OS	IBM Developer for z/OS	IBM Developer for z/OS Enterprise Edition	IBM Z and Cloud Modernization Stack (Wazi Code)
<b>Main features</b>				
3270 interface, including z/OS Debugger Utilities	√		√	
Eclipse IDE, see Table 3 on page xxiii for feature details. <sup>1</sup>	√	√	√	√
IBM Z Open Debug provided with the Wazi for Dev Spaces IDE, see Table 3 on page xxiii for feature details. <sup>1</sup>			√	√
IBM Z Open Debug provided with the Wazi for VS Code IDE, see Table 3 on page xxiii for feature details. <sup>1</sup>			√	√
<b>Code Coverage features</b>				
Compiled Language Code Coverage <sup>2</sup>	√	√ <sup>3</sup>	√	
Headless Code Coverage	√	√	√	
Java™ Code Coverage		√	√	
ZUnit Code Coverage <sup>4</sup>		√	√	
z/OS Debugger Code Coverage (3270 and remote interfaces) <sup>5</sup>	√		√	
<b>3270 features</b>				
z/OS Debugger full screen, batch or line mode	√		√	
IMS Isolation support	√		√	

Table 2. Debug feature comparison (continued)				
	IBM Debug for z/OS	IBM Developer for z/OS	IBM Developer for z/OS Enterprise Edition	IBM Z and Cloud Modernization Stack (Wazi Code)
<b>Compiler support features</b>				
Assembler support: Create EQALANGX files	√	√	√	
Assembler support: Debugging <sup>6</sup>	√	√	√ <sup>7</sup>	√ <sup>7</sup>
LANGX COBOL support <sup>8</sup>	√	√	√	
Support for Automatic Binary Optimizer (ABO)	√	√	√	
Load Module Analyzer <sup>9</sup>	√		√	

**Notes:**

- The following features are supported only in remote debug mode:
  - Support for 64-bit COBOL feature of z/OS for COBOL V6.3 and later
  - Support for 64-bit Enterprise PL/I for z/OS Version 5
  - Support for 64-bit C/C++ feature of z/OS
  - Support for IBM Open Enterprise SDK for Go 1.16.
- Code coverage does not support Go programs.
- IBM Developer for z/OS includes z/OS Debugger remote debug and compiled code coverage Eclipse interface, but does not include z/OS Debugger Code Coverage.
- ZUnit Code Coverage is only supported in Debug Tool compatibility mode.
- z/OS Debugger Code Coverage can only be enabled in the 3270 interface.
- Debugging assembler requires that you have EQALANGX files that have been created via ADFz Common Components or a product that ships the ADFz Common Components.
- This feature is only available with the Eclipse IDE.
- LANGX COBOL refers to any of the following programs:
  - A program compiled with the IBM OS/VS COBOL compiler.
  - A program compiled with the IBM VS COBOL II compiler with the NOTEST compiler option.
  - A program compiled with the IBM Enterprise COBOL for z/OS Version 3 or Version 4 compiler with the NOTEST compiler option.
- Load Module Analyzer is deprecated and will be removed in a future version.

Table 3. Remote IDE debug feature comparison		
Feature	Eclipse-based debug interface	IBM Z Open Debug <sup>1,10</sup>
Debug Tool compatibility mode <sup>2</sup>	√	√
Standard mode <sup>3,10</sup>	√ <sup>4</sup>	

Table 3. Remote IDE debug feature comparison (continued)

Feature	Eclipse-based debug interface	IBM Z Open Debug <sup>1,10</sup>
Integration with Language Editors <sup>10</sup>	<ul style="list-style-type: none"> <li>• COBOL Editor<sup>5</sup></li> <li>• PL/I Editor<sup>5</sup></li> <li>• Remote C/C++ Editor<sup>4,5</sup></li> <li>• System z LPEX Editor<sup>4,5</sup></li> </ul>	<ul style="list-style-type: none"> <li>• Z Open Editor</li> </ul>
Visual Debug	√ <sup>5,10</sup>	
Debugging ZUnit tests	√ <sup>6,10</sup>	
Debug profile management	√ <sup>4,10</sup>	√
IMS Isolation UI	√ <sup>7</sup>	
Integration with CICS Explorer views	√ <sup>4,5</sup>	
Integration with property groups	√ <sup>5,10</sup>	
Team Debug support	√ <sup>4,5</sup>	
Integrated launch <sup>10</sup>	<ul style="list-style-type: none"> <li>• z/OS UNIX Application launch configuration</li> <li>• z/OS Batch Application using existing JCL</li> <li>• z/OS Batch Application using a property group<sup>5</sup></li> </ul>	
Debug Tool Plug-ins	√ <sup>4, 8</sup>	
Modules	√	
Memory	√	
<b>Program navigation</b>		
Step over/Next	√	√
Step into/Step in	√	√
Step return/Step out	√	√
Jump to location	√ <sup>10</sup>	
Run to location/Run to cursor	√ <sup>10</sup>	√
Resume/Continue	√	√
Terminate	√	√
Animated step	√	
Playback	√ <sup>10</sup>	
<b>Breakpoints</b>		
Line/statement breakpoints	√	√
Entry breakpoints	√	
Source entry breakpoints	√ <sup>10</sup>	
Event breakpoint	√ <sup>10</sup>	



Table 3. Remote IDE debug feature comparison (continued)		
Feature	Eclipse-based debug interface	IBM Z Open Debug <sup>1,10</sup>
Address breakpoint	√ <sup>10</sup>	
Watch breakpoint	√ <sup>10</sup>	
<b>Variables &amp; Registers</b>		
Variables	√	√
Registers	√	√ <sup>9</sup>
Modifying variable and register values	√	√
Setting variable filter	√	
Changing variable representation	√	
Dereferencing variables	√	
Displaying in memory view	√	
<b>Monitors</b>		
Displaying monitor	√	√
Modifying monitor value	√	
Changing variable representation	√	
Dereferencing variables	√	
<b>Debug Console</b>		
Evaluating variables and expressions		√
z/OS Debugger commands	√ <sup>10</sup>	

**Notes:**

1. IBM Z Open Debug is provided with Wazi for Dev Spaces and Wazi for VS Code.
2. Debug Tool compatibility mode does not support 64-bit Enterprise PL/I for z/OS Version 5.
3. Standard mode does not support 64-bit COBOL feature of z/OS for COBOL V6.3 and later. Source view for COBOL V6.2 and later is supported only in standard mode.
4. This feature is not available in Wazi for Eclipse.
5. This feature is not available in IBM Debug for z/OS.
6. Debugging ZUnit tests is only supported in Debug Tool compatibility mode.
7. This feature is only available in IBM Developer for z/OS Enterprise Edition.
8. IBM Developer for z/OS includes Debug Tool plug-ins, but does not include Load Module Analyzer and z/OS Debugger Code Coverage 3270 interfaces.
9. Registers are available in the **Variables** view.
10. Programs compiled with IBM Open Enterprise SDK for Go are not supported.



## Chapter 1. z/OS Debugger commands

z/OS Debugger provides the following commands:

## ? command

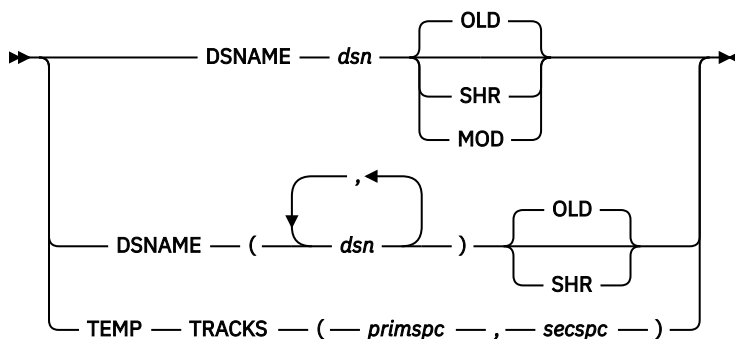
Displays a list of all commands or, if used in combination with a command, displays a list of options that you can specify for that command.

► ? — ; ◄

## 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.

➤ **ALLOCATE** — **FILE** — *ddname* — **attributes** — ; ➤

**attributes**

## ANALYZE command (PL/I)

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

►► ANALYZE — EXPRESSION — ( — *expression* — ) — ; ◄◄

## Assignment command (assembler and disassembly)

The Assignment command copies the value of an expression to a specified memory location or register.

$\Rightarrow$  receiver  $\overbrace{\hspace{10em}}^{\text{sourceexpr}} = \text{sourceexpr} \text{ ; } \Rightarrow$   
 $\hspace{10em} \underbrace{\hspace{10em}}_{\text{receiverlen}}$

## Assignment command (LangX 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 z/OS Debugger control when storage for a named controlled variable or aggregate is dynamically allocated by PL/I.

➤ AT — *every\_clause* — ALLOCATE — *identifier* — *command* — ; ➤

( *identifier* )

\*

## AT APPEARANCE

Gives z/OS Debugger control when the specified compile unit is found in storage.

➤ AT — *every\_clause* — APPEARANCE — *cu\_spec* — *command* →

( *cu\_spec* )

\*

➤ ; ➤

## AT CALL

Gives z/OS Debugger control when the application code attempts to call the specified entry point.

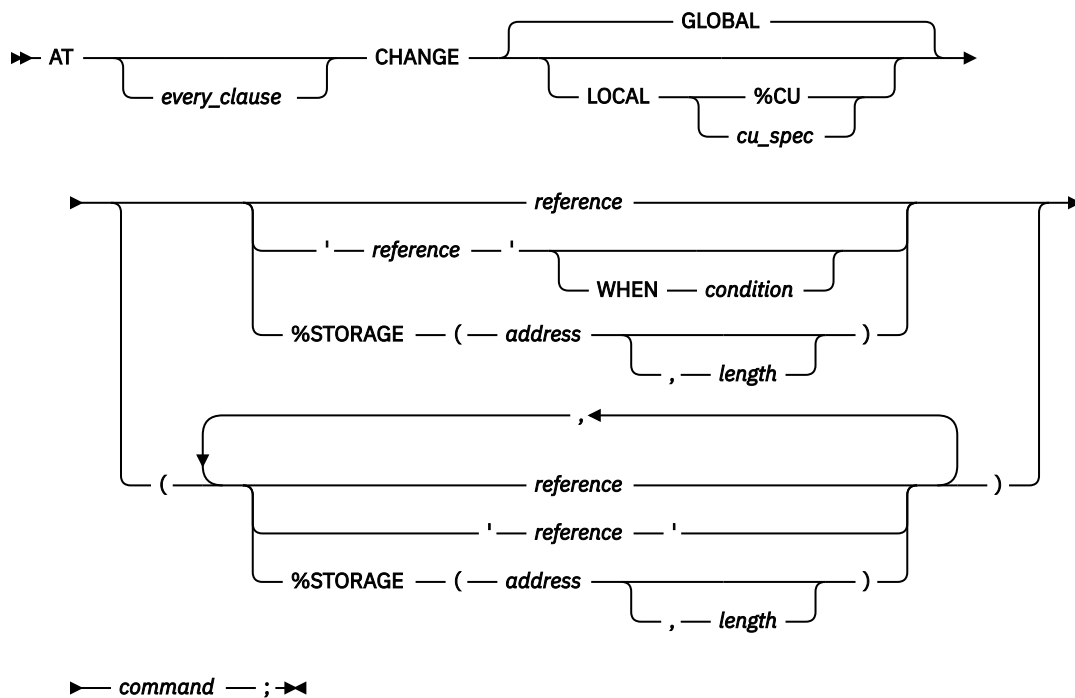
➤ AT — *every\_clause* — CALL — *entry\_name* — *command* — ; ➤

( *entry\_name* )

\*

## AT CHANGE

Gives z/OS Debugger control when either the program or z/OS Debugger command changes the specified variable value or storage location, and, optionally, when the specified condition is met.



## AT CHANGE (remote)

Gives z/OS Debugger control when either the program or z/OS Debugger command changes the specified variable value.

```

AT CHANGE "reference" ;
'reference'

```

## AT CURSOR (full-screen mode)

Provides a cursor controlled method for setting a statement breakpoint.

```

AT CURSOR ;
TOGGLE

```

## AT DATE (COBOL)

Gives z/OS Debugger control for each date processing statement within the specified block.

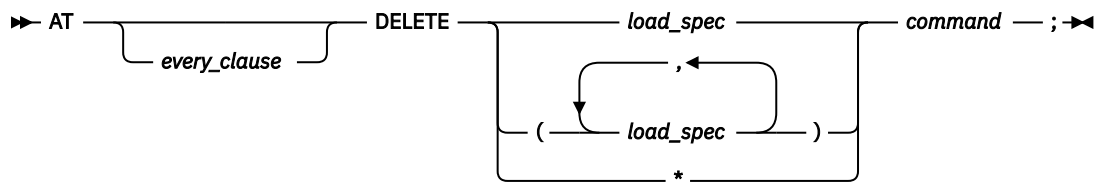
```

AT DATE block_spec command ;
every_clause
(
  block_spec
  *
)

```

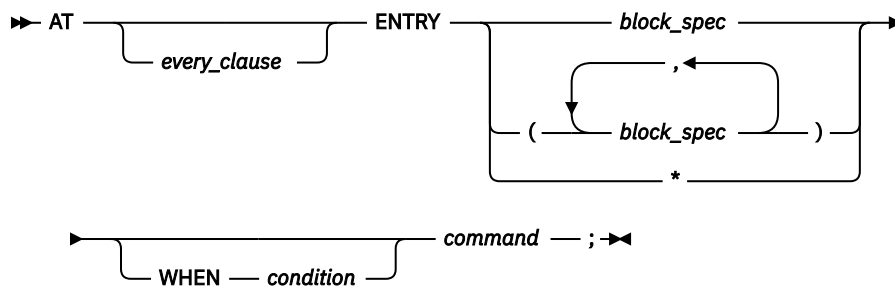
## AT DELETE

Gives z/OS Debugger 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.



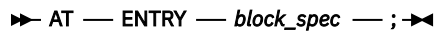
## AT ENTRY

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



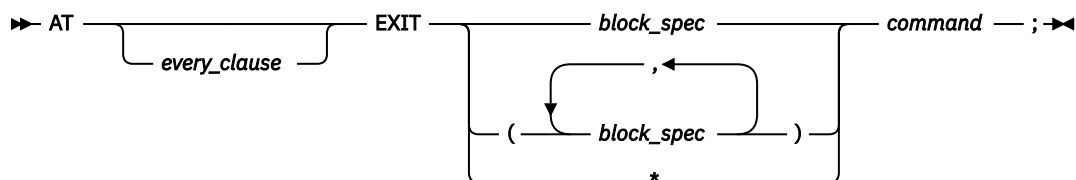
## AT ENTRY (remote)

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



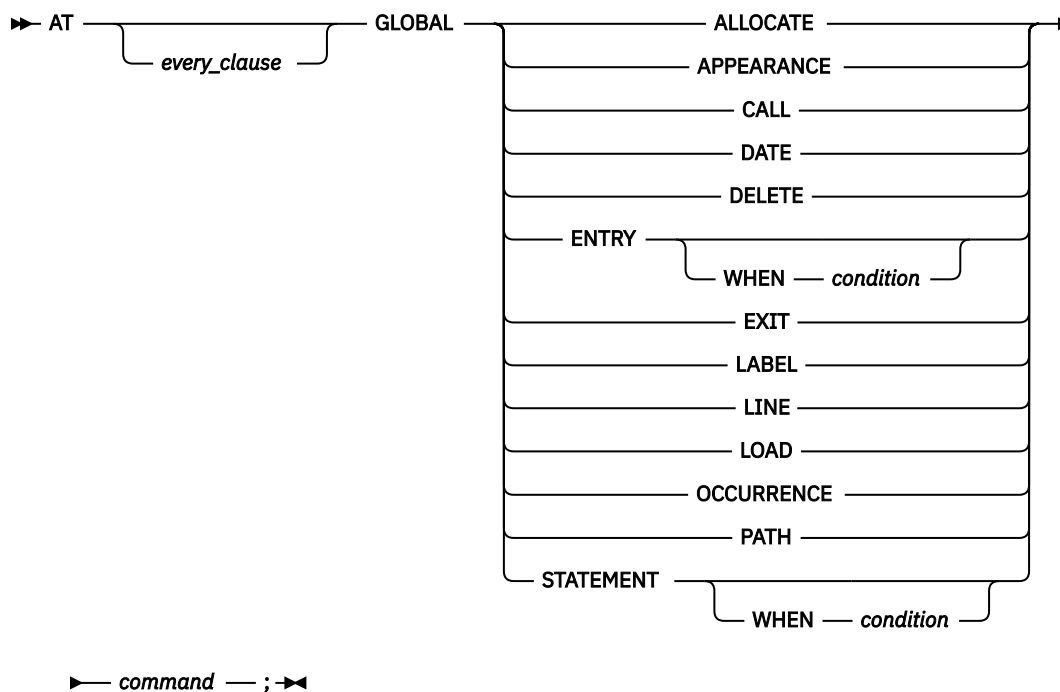
## AT EXIT

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



## AT GLOBAL

Gives z/OS Debugger control for every instance of the specified AT-condition.



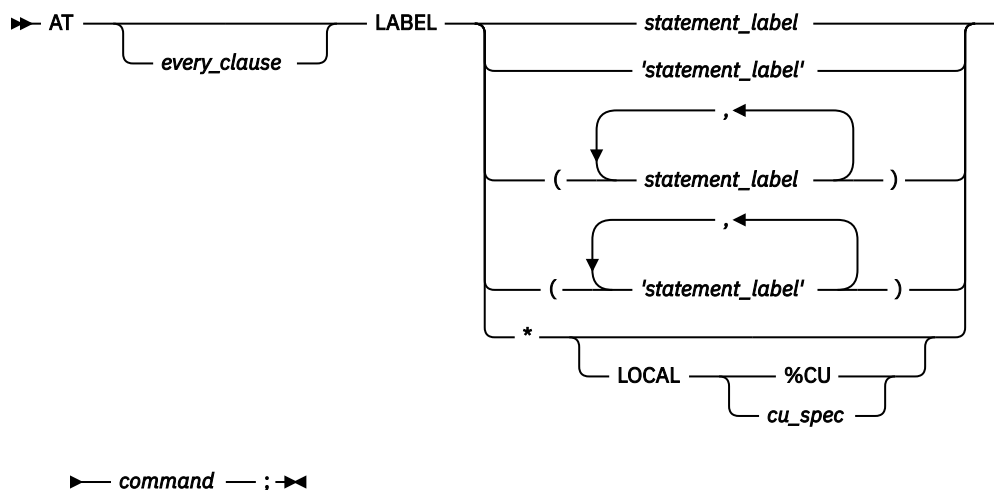
## AT GLOBAL LABEL (remote)

Gives z/OS Debugger control for every instance of the specified AT-Label condition.

►► AT — GLOBAL — LABEL — ; ►►

## AT LABEL

Gives z/OS Debugger control when execution has reached the specified statement label or group of labels.



## AT LABEL (remote)

Gives z/OS Debugger control when execution reaches the statement label that you specify.

➤ AT LABEL *statement\_label* ; ➤

└──────────┴──────────┘

\*

## AT LINE

Gives z/OS Debugger control at the specified line. See “AT STATEMENT” on page 7.

## AT LOAD

Gives z/OS Debugger control when the specified load module is brought into storage.

➤ AT *every\_clause* LOAD *module\_name* ; ➤

└──────────┴──────────┘

,

└──────────┴──────────┘

( *module\_name* )

└──────────┴──────────┘

,

└──────────┴──────────┘

( *load\_spec* )

\*

└──────────┴──────────┘

command ; ➤

## AT LOAD (remote)

Gives z/OS Debugger control when the specified load module is brought into storage.

➤ AT — LOAD — *module\_name* — ; ➤

## AT OCCURRENCE

Gives z/OS Debugger control on a language or Language Environment condition or exception.

➤ AT *every\_clause* OCCURRENCE *condition* *command* ➤

└──────────┴──────────┘

,

└──────────┴──────────┘

( *condition* )

└──────────┴──────────┘

;

## AT OFFSET (disassembly)

Gives z/OS Debugger control at the specified offset in the disassembly view.

➤ AT — OFFSET *offset\_spec* *command* ; ➤

└──────────┴──────────┘

,

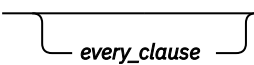
└──────────┴──────────┘

( *offset\_spec* )



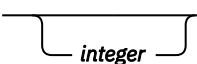
## AT PATH

Gives z/OS Debugger control when the flow of control changes (at a path point). AT PATH is identical to AT GLOBAL PATH.

➤ AT  PATH — *command* — ; ➤

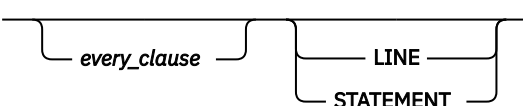
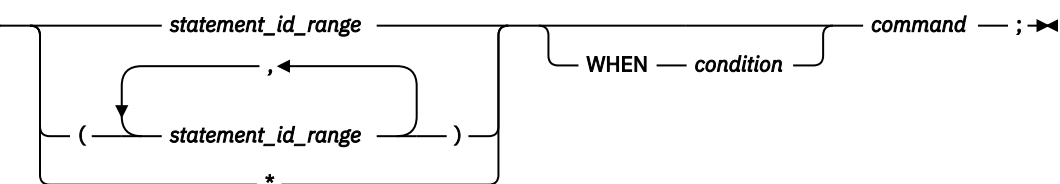
## AT Prefix (full-screen mode)

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

➤ AT  ; ➤

## AT STATEMENT

Gives z/OS Debugger control at each specified statement or line within the given set of ranges, and, optionally, when the specified condition is met.

➤ AT   WHEN — *condition* — *command* — ; ➤

## AT STATEMENT (remote)

Gives z/OS Debugger control at each specified statement or line.

➤ AT  *statement\_id* — ; ➤


## AT TERMINATION

Gives z/OS Debugger control when the application program is terminated.

➤ AT — TERMINATION — *command* — ; ➤

## BEGIN command (programming language neutral)

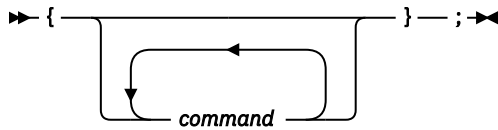
BEGIN and END delimit a sequence of one or more commands to form one longer command.

➤ BEGIN — ;  END — ; ➤

## block command (C and C++)

---

The block command allows you to group any number of z/OS Debugger commands into one 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 — %CECI — ; ->>

## CALL %DUMP

---

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

>> CALL — %DUMP — ( — *options\_string* — , — *title* — ) — ; ->>

## CALL %FA

---

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

>> CALL — %FA — ; ->>

## CALL %FM

---

Starts IBM File Manager for z/OS.

>> CALL — %FM — *userID* — BACKGROUND — ; ->>

## CALL %HOGAN

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

►► CALL — %HOGAN — ; ►◄

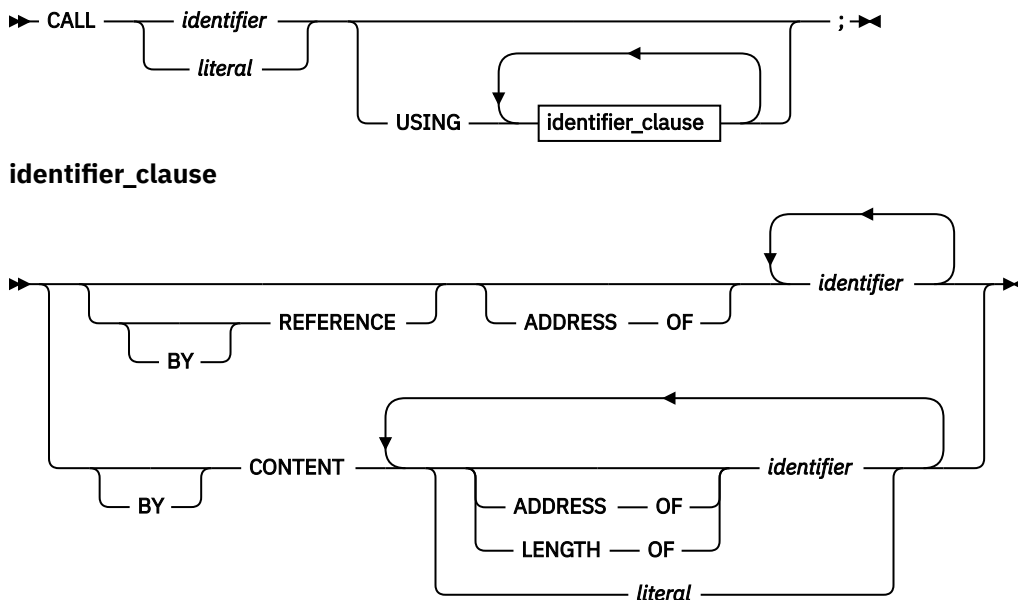
## CALL %VER

Adds a line to the log describing the maintenance level of z/OS Debugger that you have installed on your system.

►► CALL — %VER — ; ►◄

## CALL entry\_name (COBOL)

Calls an entry name in the application program.



## CALL procedure

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

►► CALL — *procedure\_name* — ; ►◄

## CC command

Controls whether code coverage data is collected.

►► CC — START — ; ►◄  
          STOP —

## CHKSTGV

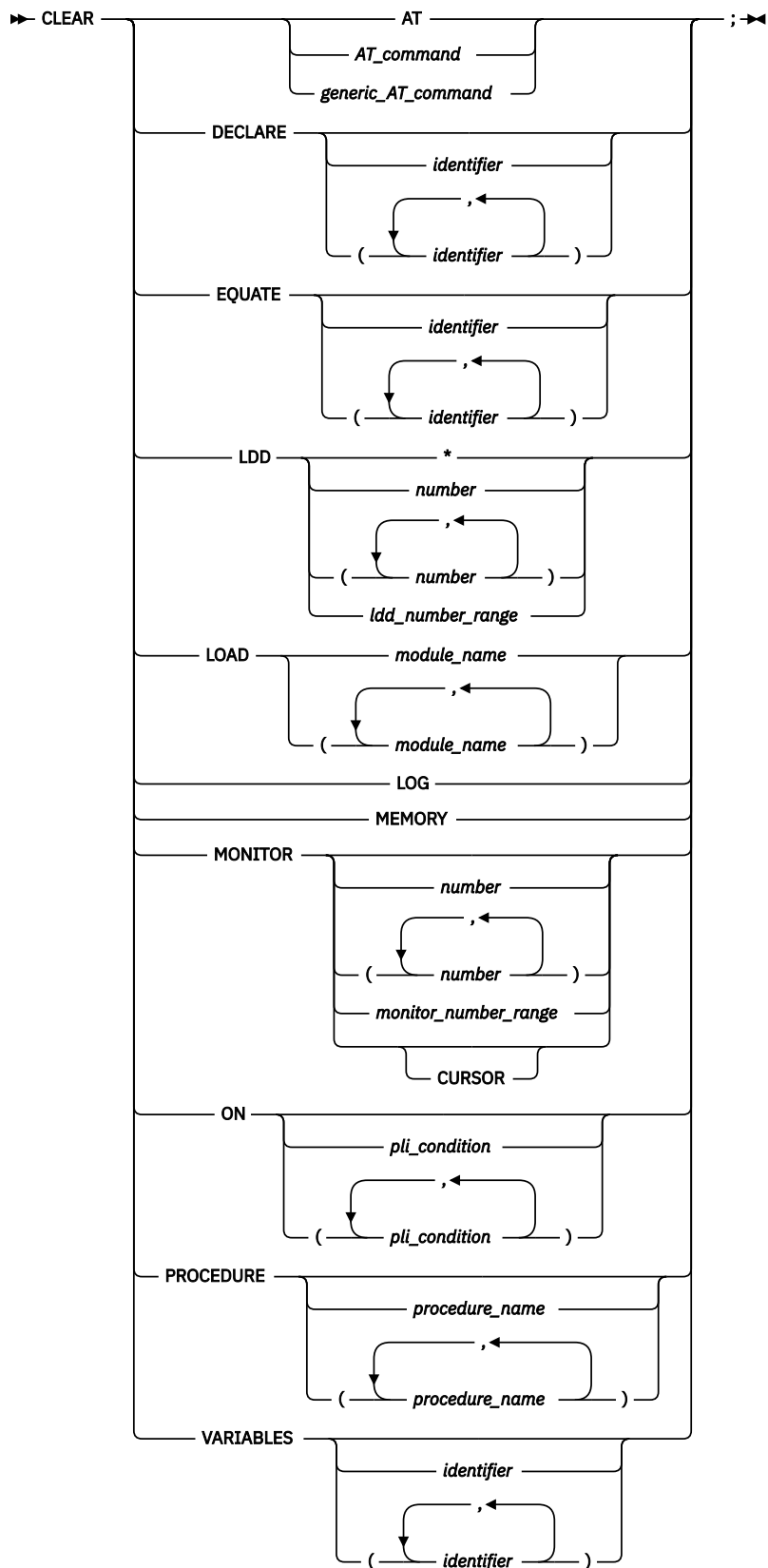
Check for a specific type of storage violation in the task you are currently debugging.

➤ CHKSTGV — ; ➤

## CLEAR command

---

The CLEAR command removes the actions of previously issued z/OS Debugger commands.



## CLEAR AT (remote)

You can use the **CLEAR AT** command to remove actions that were completed by using the **AT GLOBAL LABEL** or the **AT LABEL** commands.

➤➤ CLEAR AT — GLOBAL — LABEL — ; ➤➤  
 { LABEL — *statement\_label* — }  
 \*

## 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 in to 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 z/OS Debugger session panel (excluding the header fields) and the command line.

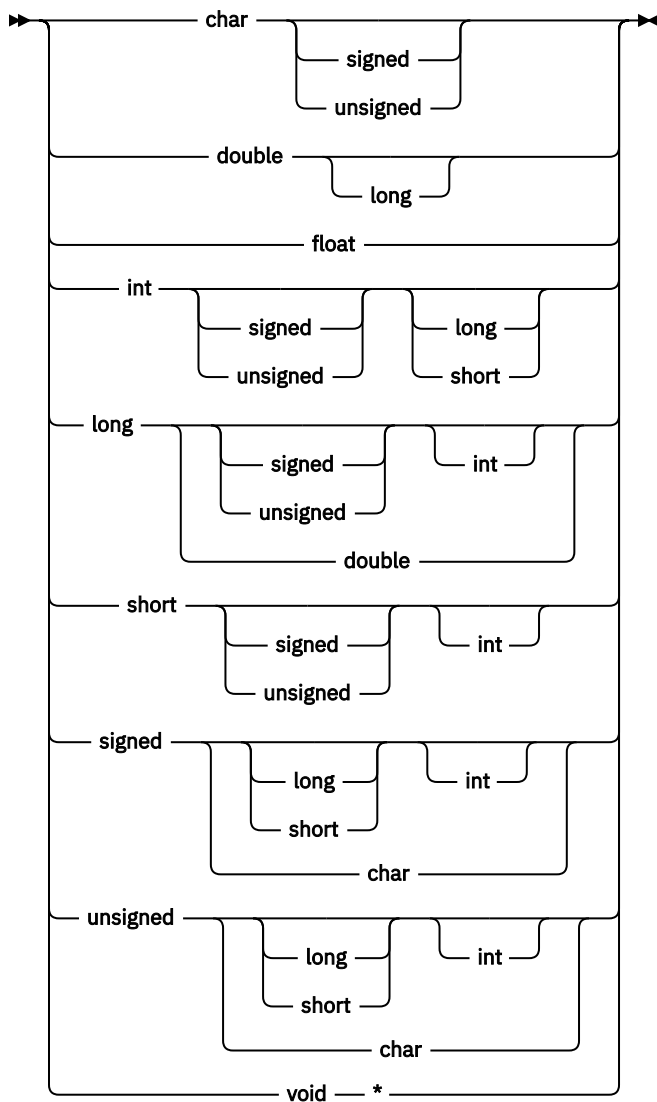
➤➤ CURSOR — ; ➤➤

## Declarations (assembler, disassembly, and LangX COBOL)

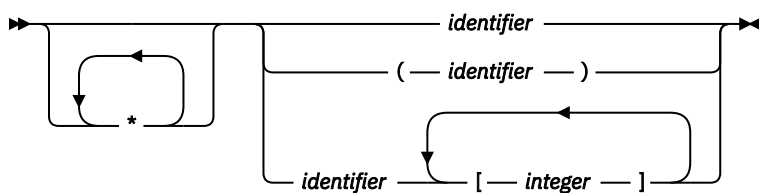
---

Use declarations to create session variables and tags effective during a z/OS Debugger session.

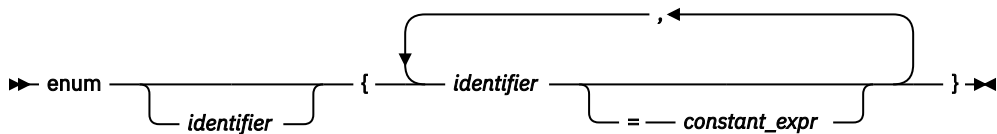




#### declarator

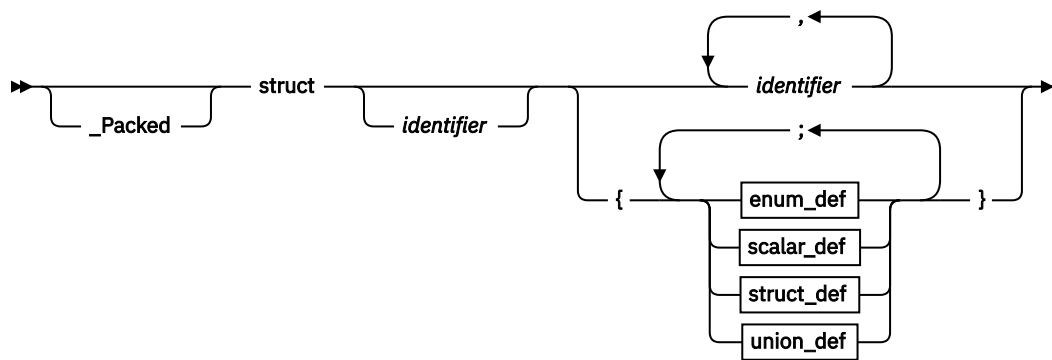


#### enum\_def

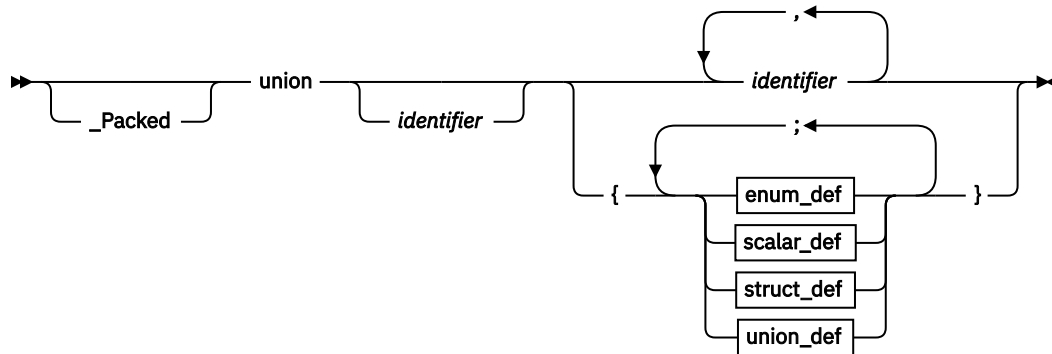


#### struct\_def



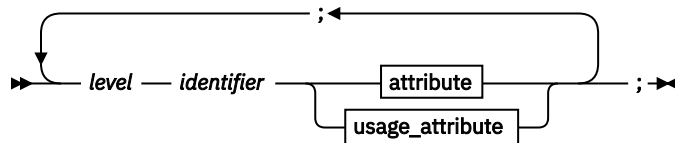


#### union\_def

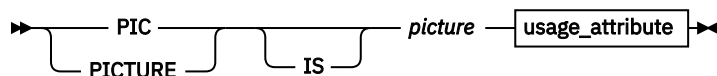


## Declarations (COBOL)

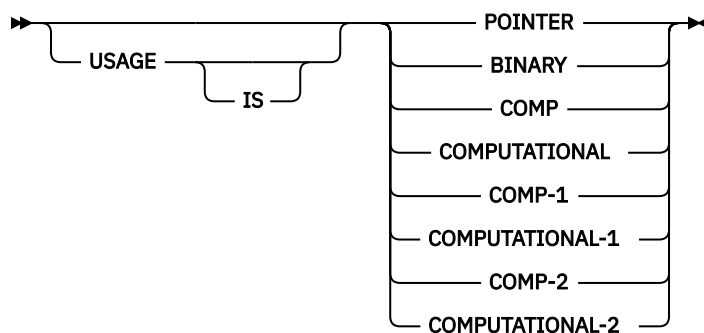
Use declarations to create session variables effective during a z/OS Debugger session.



#### attribute

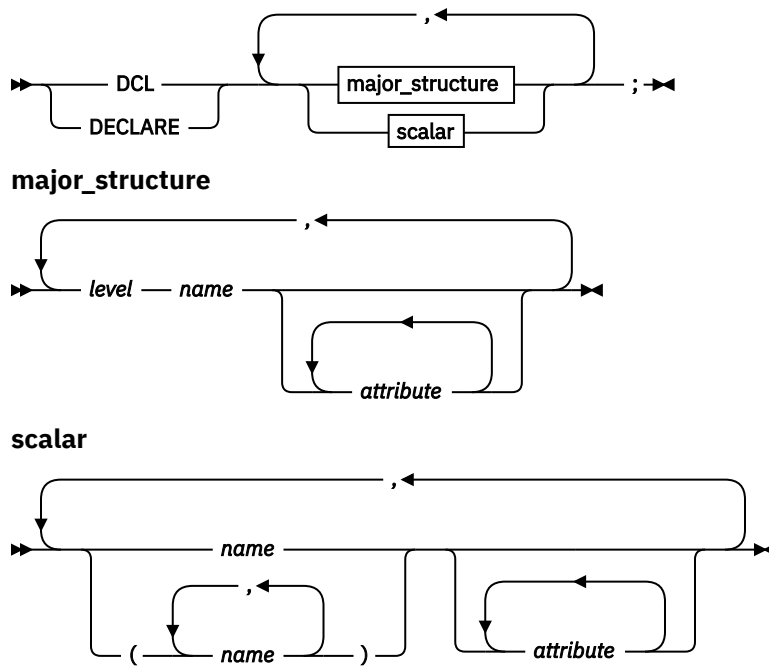


#### usage\_attribute



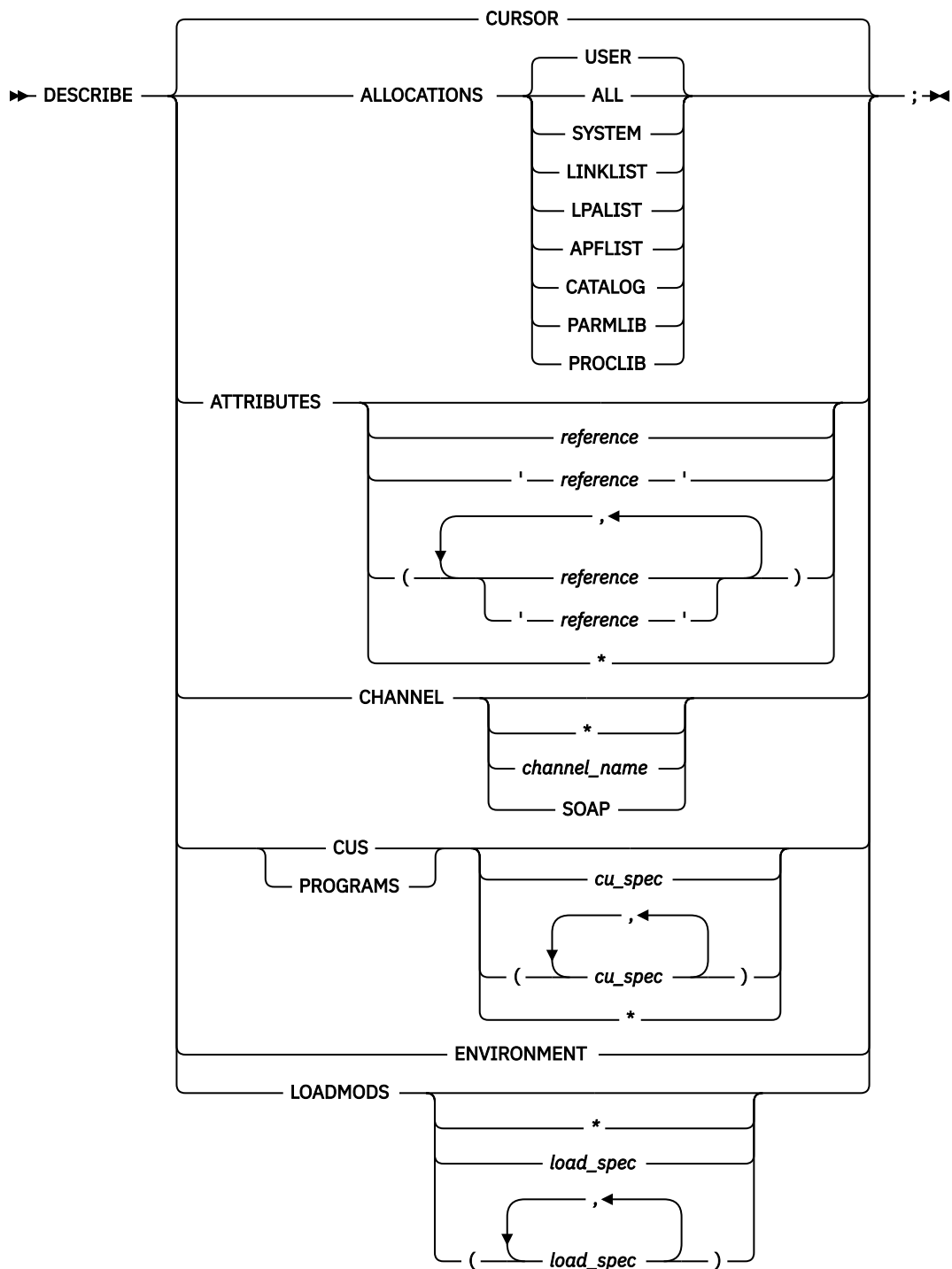
## DECLARE command (PL/I)

The `DECLARE` command creates session variables effective during a z/OS Debugger session.



## DESCRIBE command

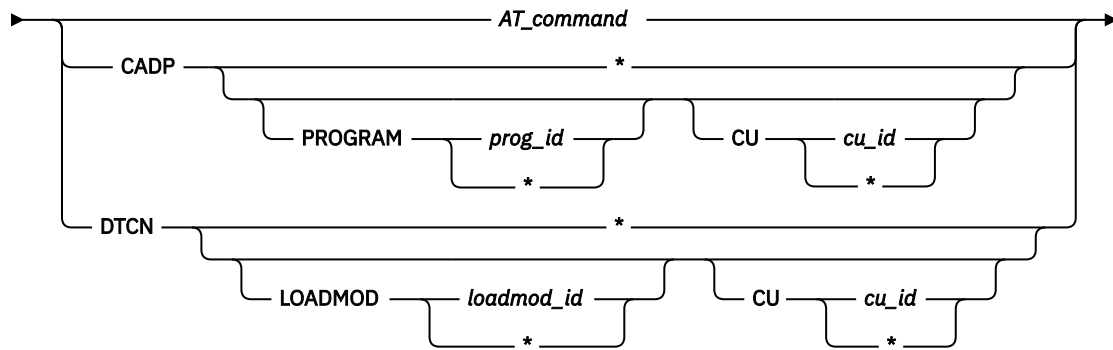
The DESCRIBE command displays the file allocations or attributes of references, compile units, known load modules, and the runtime environment.



## DISABLE command

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

➤➤ DISABLE ➡



➡ ; ➤➤

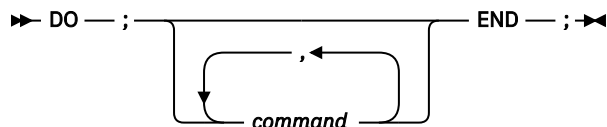
## DISABLE prefix (full-screen mode)

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

➤➤ DISABLE integer ; ➤➤

## DO command (assembler, disassembly, LangX COBOL, and COBOL)

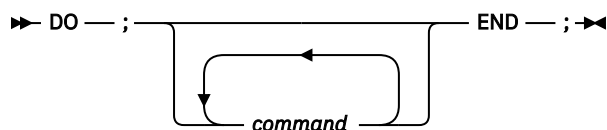
The DO command performs one or more commands that are collected into a group.



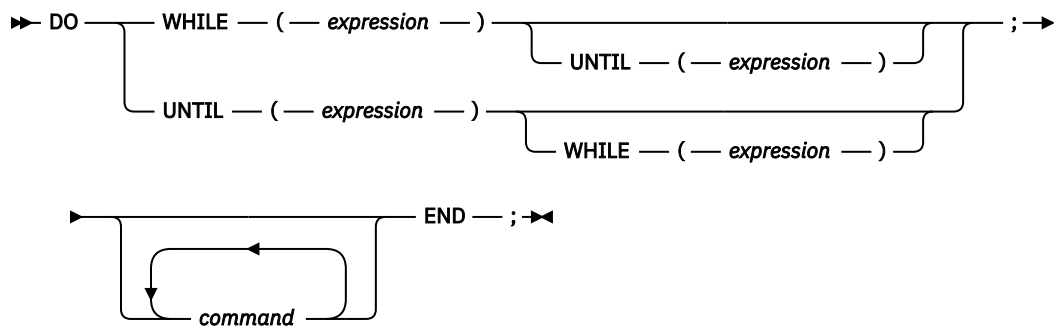
## DO command (PL/I)

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

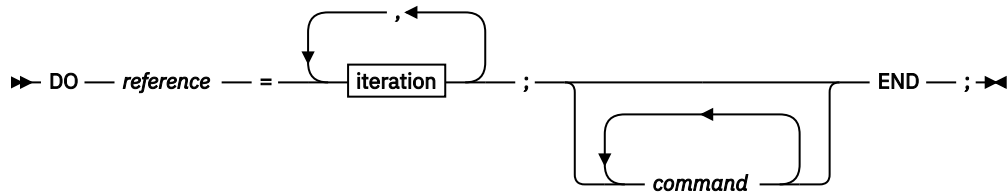
### Simple



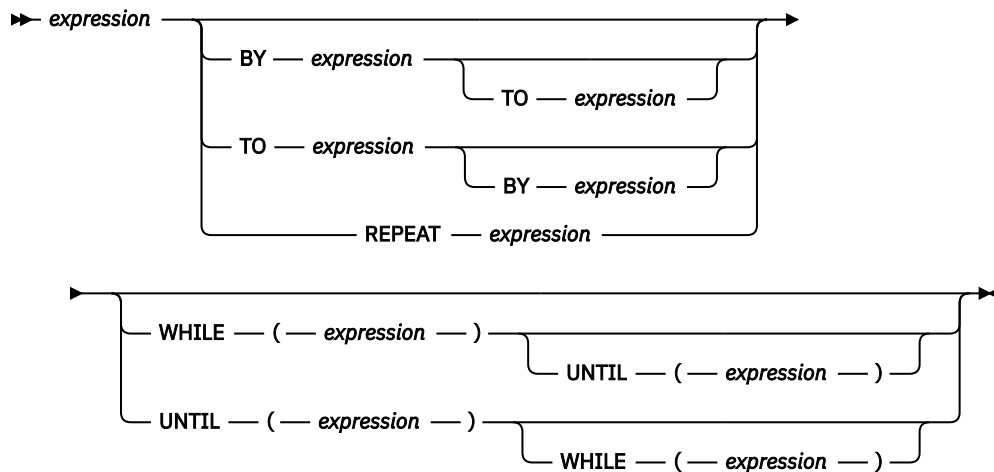
### Repeating



## Iterative



## iteration



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

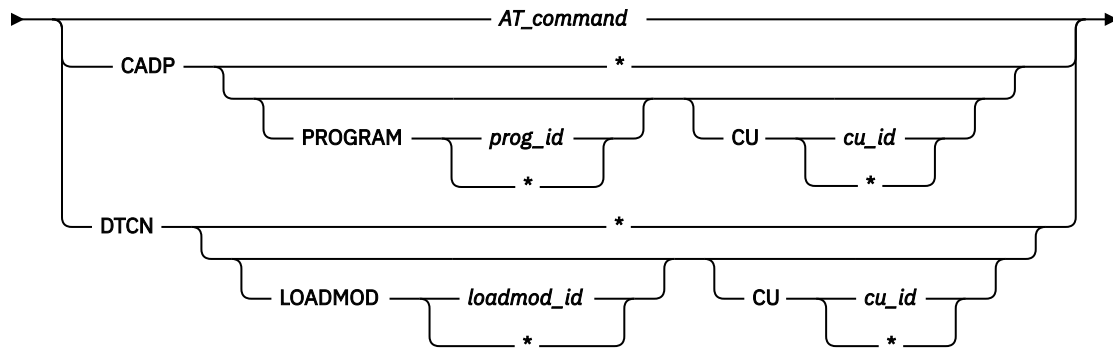
The do/while command performs a command before evaluating the test expression.

```
do command while ( — expression — ) ;
```

## ENABLE command

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

➤➤ ENABLE ➡



➡ ; ➤➤

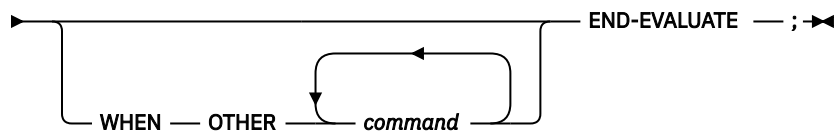
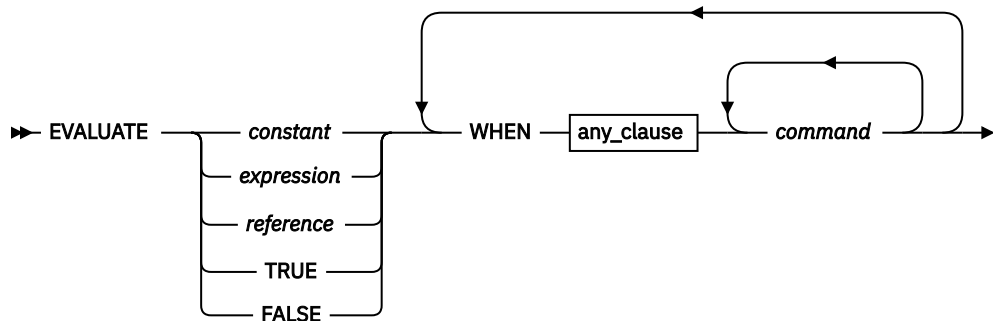
## ENABLE prefix (full-screen mode)

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

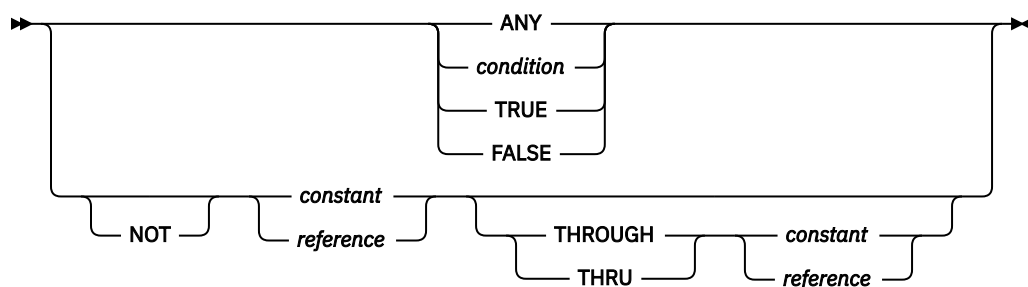


## EVALUATE command (COBOL)

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



any\_clause



## Expression command (C and C++)

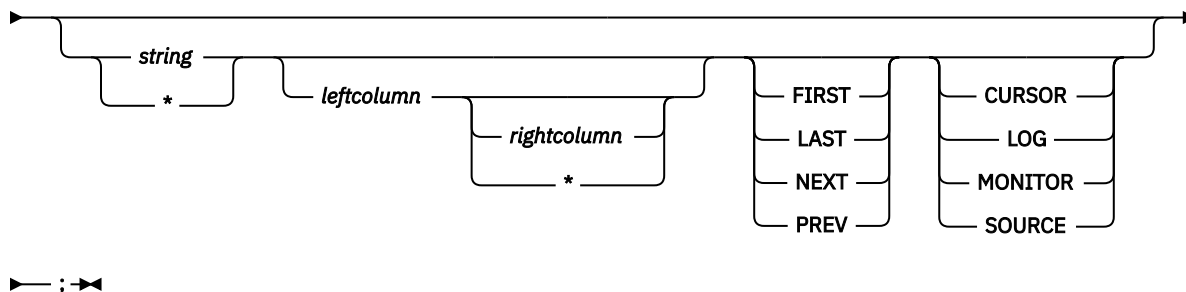
The Expression command evaluates the given expression.

➤ *expression* — ; ➤

## FIND command

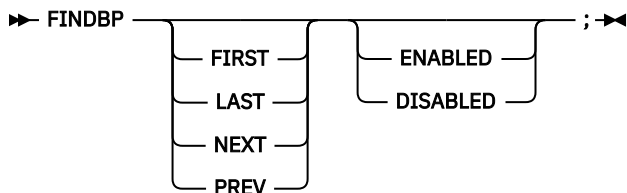
The FIND command helps you search through the Source window in full-screen and batch mode, and through the Log and Monitor windows in full-screen mode.

➤ FIND →



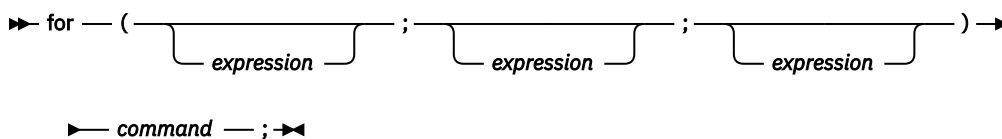
## FINDBP command

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.



## FREE command

The FREE command releases a file that is currently allocated.

➤ FREE — FILE — *ddname* — ; ➤

## GO command

The GO command causes z/OS Debugger to start or resume running your program.

➤➤ GO \_\_\_\_\_ ; ➤➤  
           └─ BYPASS ─┘

## GOTO command

The GOTO command causes z/OS Debugger to resume program execution at the specified statement id.

➤➤ GOTO \_\_\_\_\_ *statement\_id* — ; ➤➤  
      └─ GO — TO ─┘

## GOTO LABEL command

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

➤➤ GOTO \_\_\_\_\_ *statement\_label* \_\_\_\_\_ ; ➤➤  
      └─ GO — TO ─┘   └─ 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 LangX COBOL)

The IF command lets you conditionally perform a command.

➤➤ IF \_\_\_\_\_ *condition* \_\_\_\_\_ THEN — *command* \_\_\_\_\_ ; ➤➤  
      └─ ' — *condition* — ' ─┘                                   └─ ELSE — *command* ─┘

## if command (C and C++)

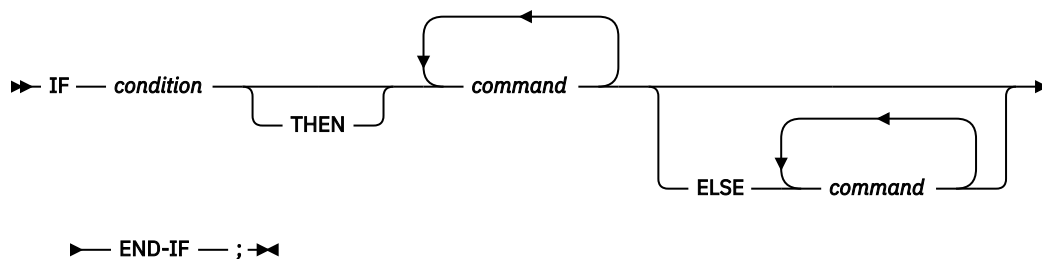
The if command lets you conditionally perform a command.

➤➤ if — ( — *expression* — ) — *command* \_\_\_\_\_ ; ➤➤  
    └─ else — *command* ─┘

## IF command (COBOL)

The IF command lets you conditionally perform a command.





## IF 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 — statement_label — ;

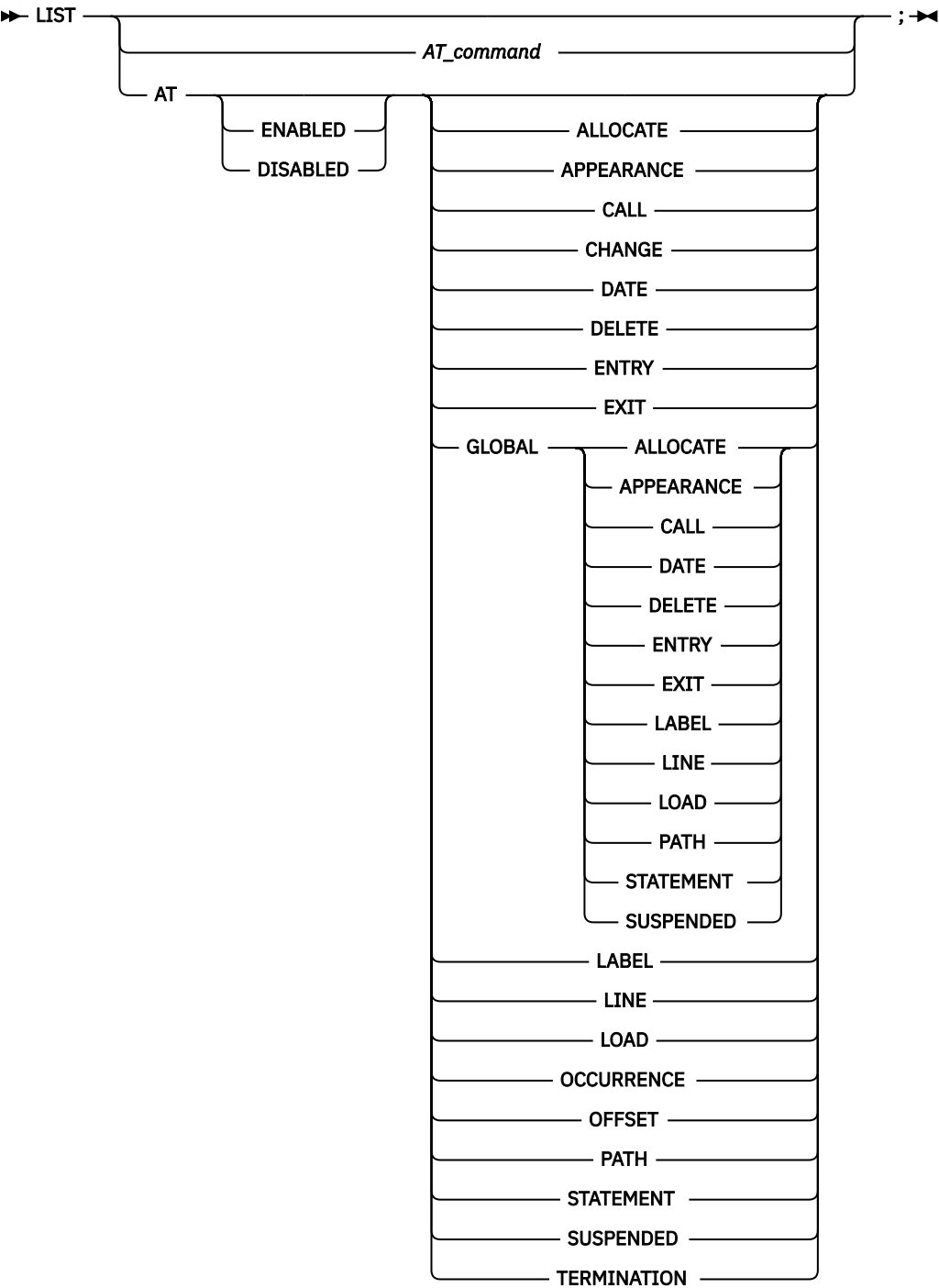
```

## 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

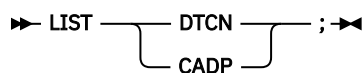
Lists the currently defined breakpoints, including the action taken when the specified breakpoint is activated.



## LIST AT (remote)

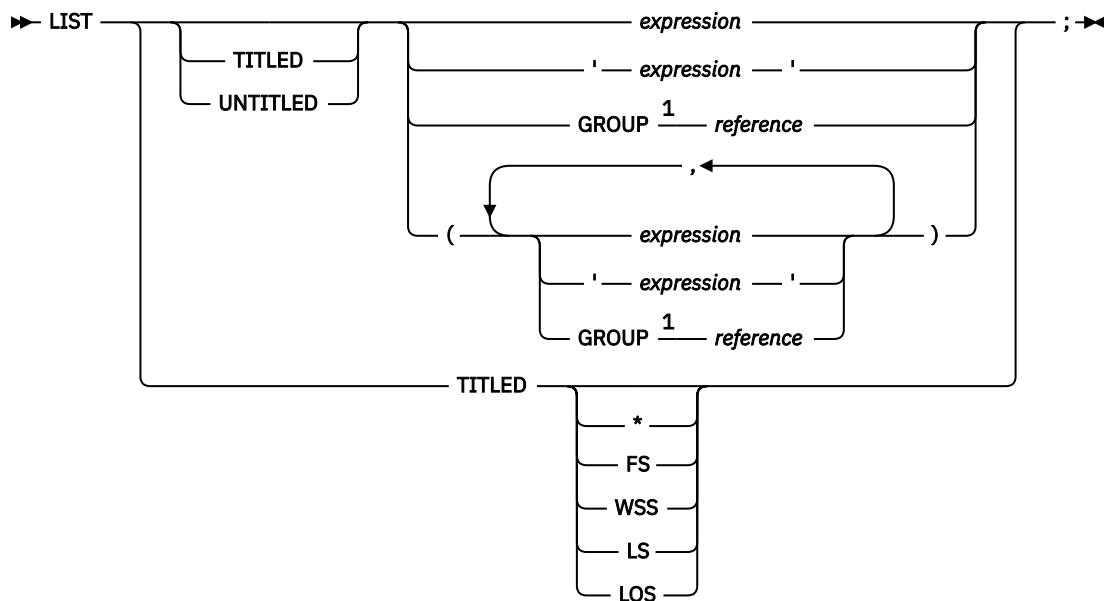
Lists the currently defined AT GLOBAL LABEL or AT LABEL breakpoints.





## LIST expression

Displays values of expressions.

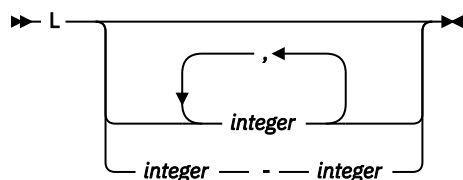


Notes:

<sup>1</sup> Only for COBOL.

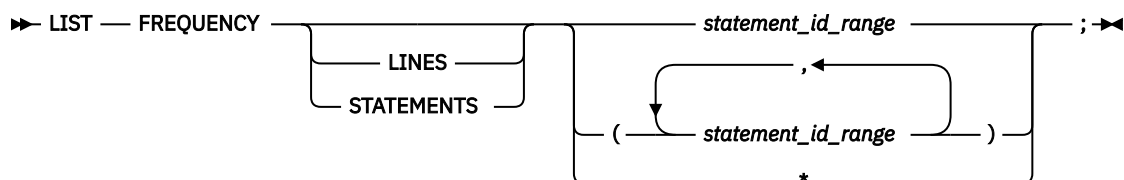
## L expression prefix

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



## LIST FREQUENCY

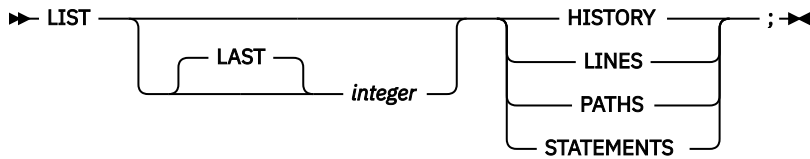
Lists statement execution counts.



## LIST LAST

---

Displays a list of recent entries in the history table.



## LIST LDD

---

Displays the list of LDD commands known to z/OS Debugger.

➤ `LIST — LDD — ;` ➤

## LIST LINE NUMBERS

---

Equivalent to `LIST STATEMENT NUMBERS`.

## LIST LINES

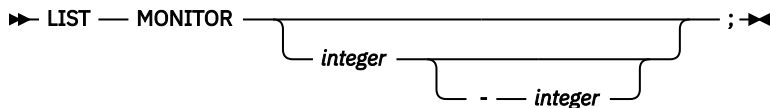
---

Equivalent to `LIST STATEMENTS`.

## LIST MONITOR

---

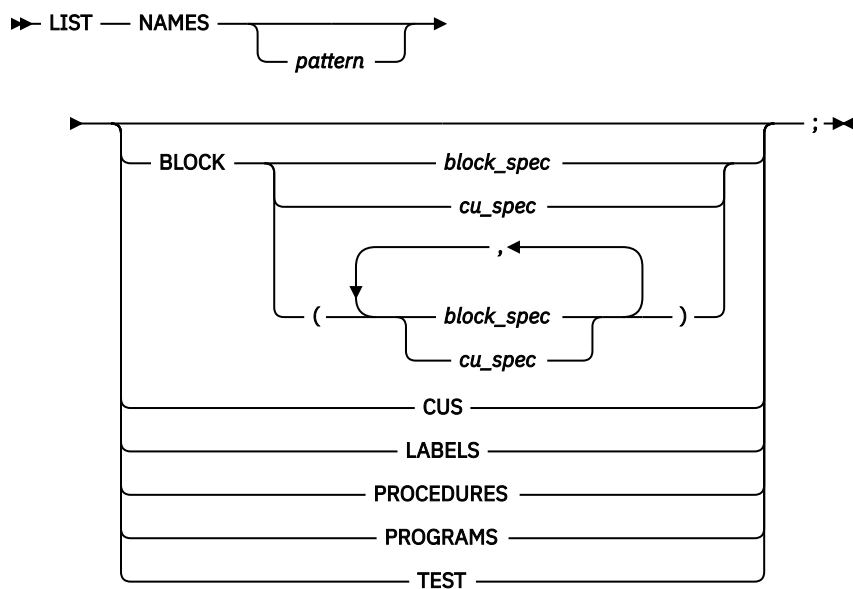
Lists all or selected members of the current set of MONITOR commands.



## LIST NAMES

---

Lists the names of variables, programs, or z/OS Debugger procedures.



## LIST NAMES LABELS (remote)

Displays the names of all section and paragraph names in a COBOL program, and the names of all instruction labels in an assembler program.

➤ LIST — NAMES — LABELS — ; ➤

## 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 z/OS Debugger PROCEDURE definitions.

➤ LIST — PROCEDURES — *name* — ; ➤

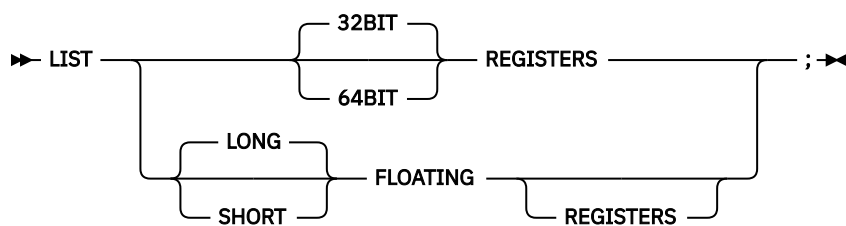
```

[
  name
  (
    name
    ,
    name
  )
] ;

```

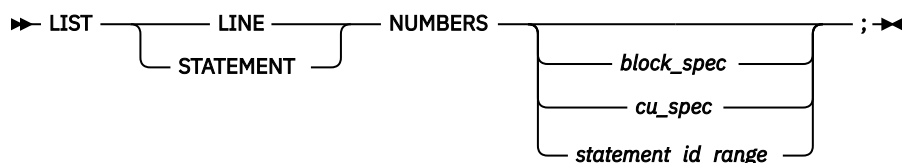
## LIST REGISTERS

Displays the current register contents.



## LIST STATEMENT NUMBERS

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



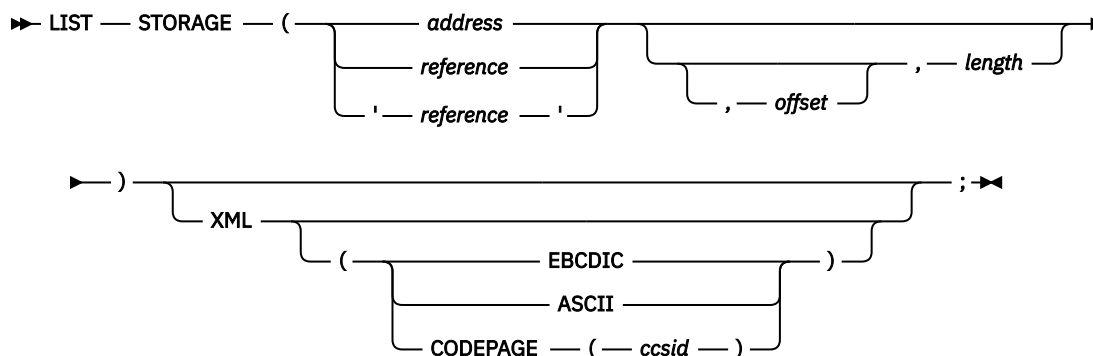
## LIST STATEMENTS

Lists one or more statements or lines from a file.



## LIST STORAGE

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



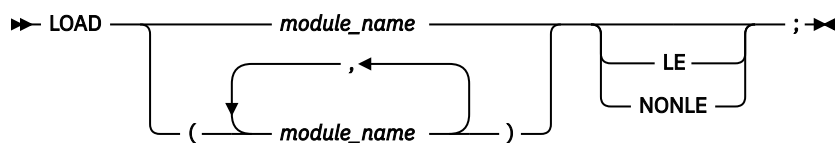
## LIST TRACE LOAD command

Displays the entries in the TRACE LOAD table that is created since the TRACE LOAD START command was issued.



## 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.

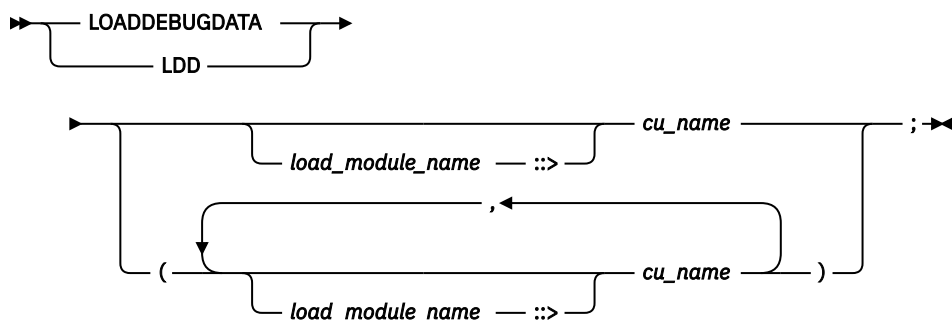


## LOADDEBUGDATA (LDD)

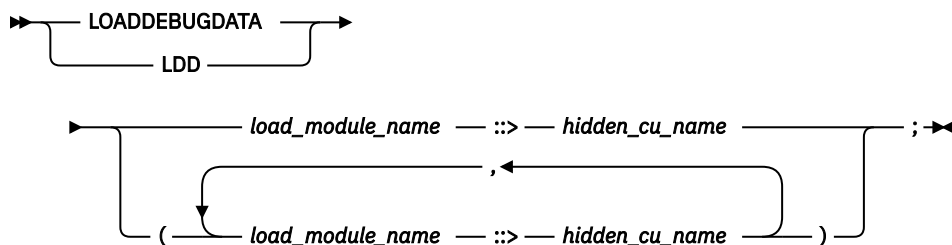
The LOADDEBUGDATA (LDD) command requests that z/OS Debugger load the debug data for a compile unit in either of the following situations:

- The CU was written in assembler or LangX COBOL.
- Explicit debug mode is active and the CU was written in a Language Environment enabled, high-level language and compiled with the TEST or DEBUG compiler option.

### LDD for assembler or LangX COBOL CU

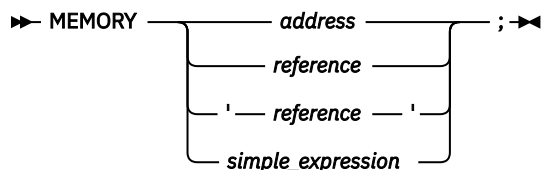


### LDD for high-level language CU in explicit debug mode



## MEMORY

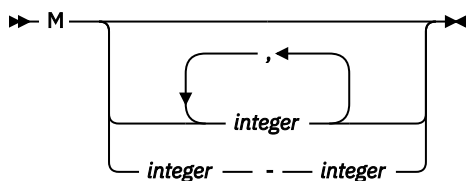
Identifies an address in memory and then display the contents of memory at that location in the Memory window. The Memory window displays memory in dump format.



## 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.

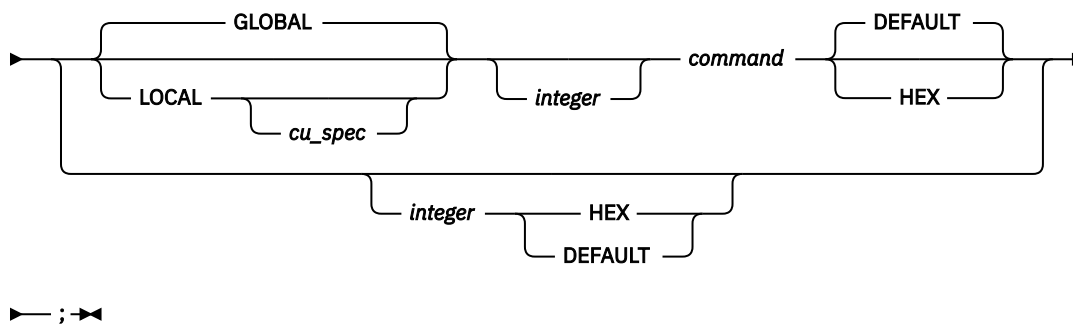




## MONITOR command

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 ➔



## MOVE command (COBOL)

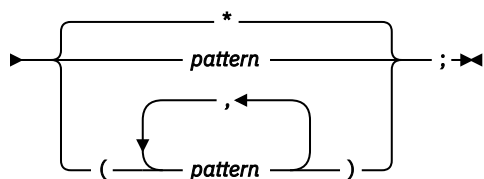
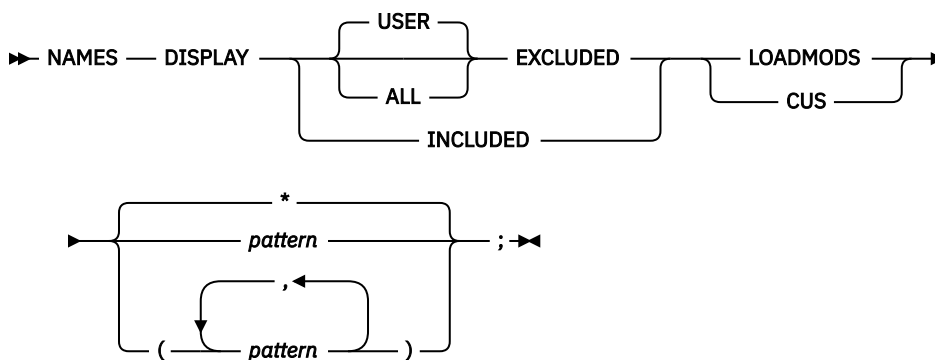
The MOVE command transfers data from one area of storage to another.

➤ MOVE *reference* TO *reference* ; ➤

*literal*

## NAMES DISPLAY command

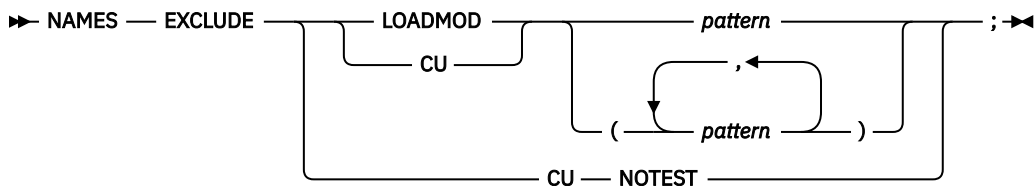
Use the NAMES DISPLAY command to indicate that you want a list of all the load modules or compile units that are currently excluded or included. If you do not specify the ALL parameter, only the names excluded by user commands appear in the list that is displayed. Names that z/OS Debugger excludes by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



## NAMES EXCLUDE command

---

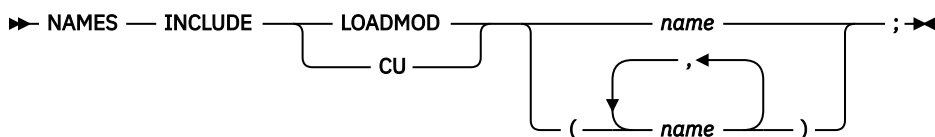
The NAMES EXCLUDE command enables you to indicate to z/OS Debugger the names of load modules or compile units that you do not need to debug. If these are data-only modules, z/OS Debugger does not process them. If they contain executable code, z/OS Debugger might process them in some cases. See "Optimizing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



## NAMES INCLUDE command

---

Use the NAMES INCLUDE command to indicate to z/OS Debugger 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 the IBM z/OS Debugger User's Guide for more information.



## Null command

---

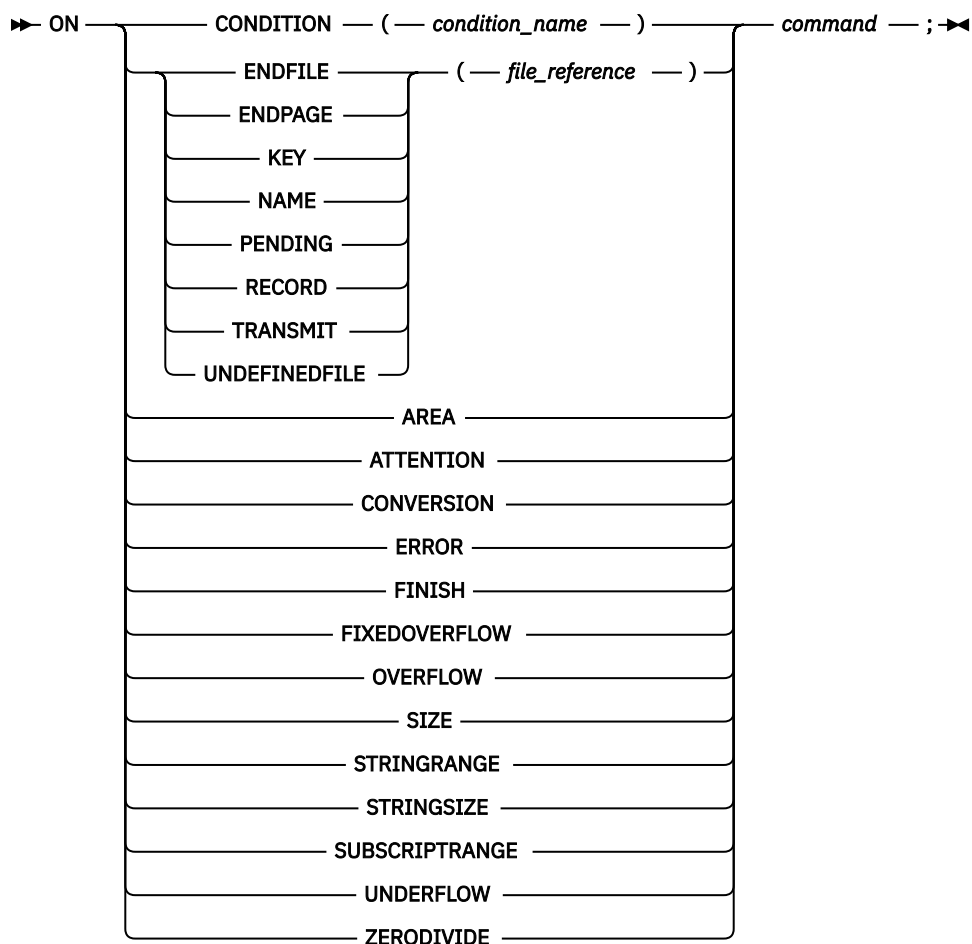
The Null command is a semicolon written where a command is expected.

➤ ; ➤

## ON command (PL/I)

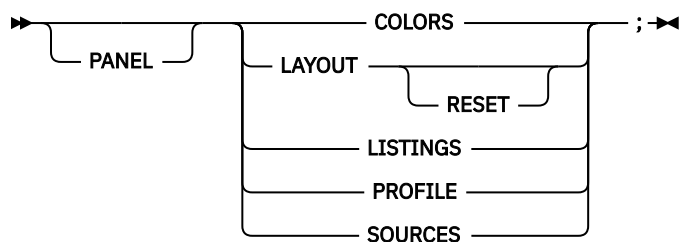
---

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



## PANEL command (full-screen mode)

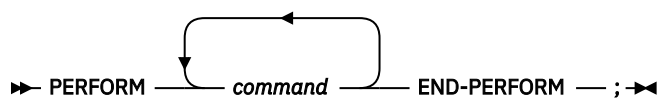
The PANEL command displays special panels.



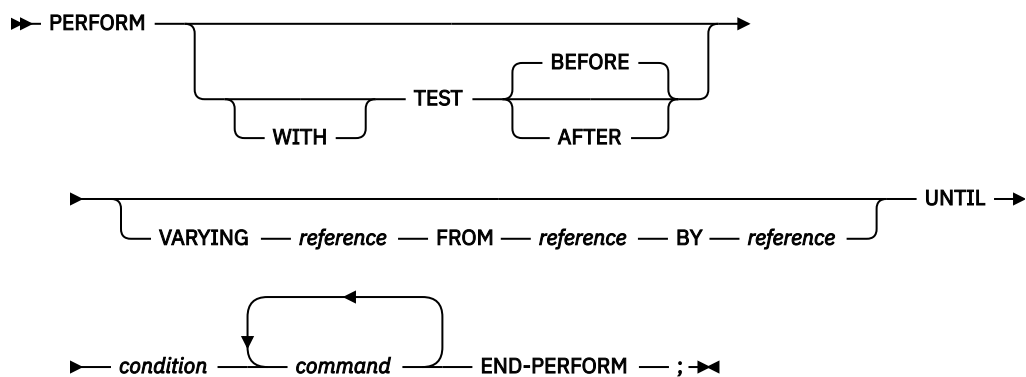
## PERFORM command (COBOL)

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:**



**Repeating:**



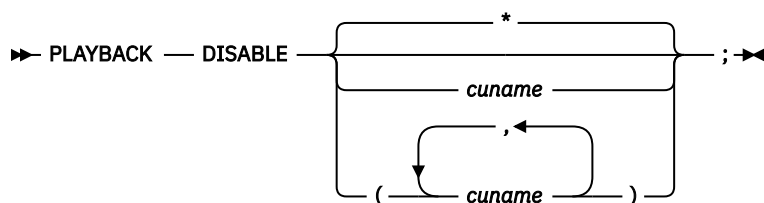
## PLAYBACK BACKWARD command

The PLAYBACK BACKWARD command indicates to z/OS Debugger to perform STEP and RUNTO commands backward, starting from the current point and going to previous points.

➤ PLAYBACK — BACKWARD — ;

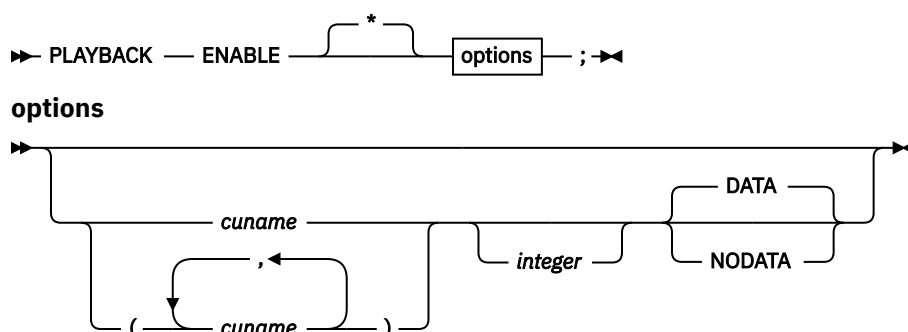
## PLAYBACK DISABLE command

The PLAYBACK DISABLE command informs z/OS Debugger to stop recording runtime environment information and discard any information recorded thus far.



## PLAYBACK ENABLE command

The PLAYBACK ENABLE command informs z/OS Debugger to begin recording the application runtime environment information (steps history and data history).



## PLAYBACK FORWARD command

The PLAYBACK FORWARD command indicates to z/OS Debugger 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 z/OS Debugger 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.

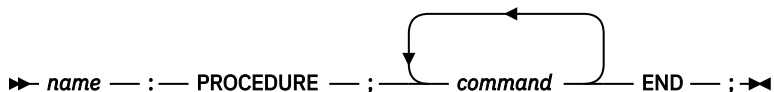
The following table summarizes the forms of the Prefix commands.

Example	Description
"AT Prefix (full-screen mode)" on page 7	Defines a statement breakpoint through the Source window prefix area.
"CLEAR prefix (full-screen mode)" on page 12	Clears a breakpoint through the Source window prefix area.
"DISABLE prefix (full-screen mode)" on page 18	Disables a breakpoint through the Source window prefix area.

Example	Description
<a href="#">"ENABLE prefix (full-screen mode)" on page 20</a>	Enables a disabled breakpoint through the Source window prefix area.
<a href="#">"LIST expression" on page 26</a>	Displays the value of a variable or variables on that line.
<a href="#">"QUERY prefix (full-screen mode)" on page 39</a>	Queries what statements have breakpoints through the Source window prefix area.
<a href="#">"RUNTO prefix command (full-screen mode)" on page 40</a>	Runs the program to the location that the cursor or statement identifier indicate in the Source window prefix area.
<a href="#">"SHOW prefix command (full-screen mode)" on page 52</a>	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.

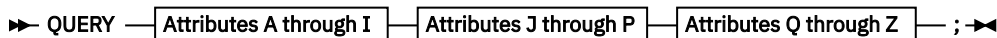

**name** — : — **PROCEDURE** — ; — **command** — **END** — ; —

## QUALIFY RESET

This command is equivalent to SET QUALIFY RESET.

## QUERY command

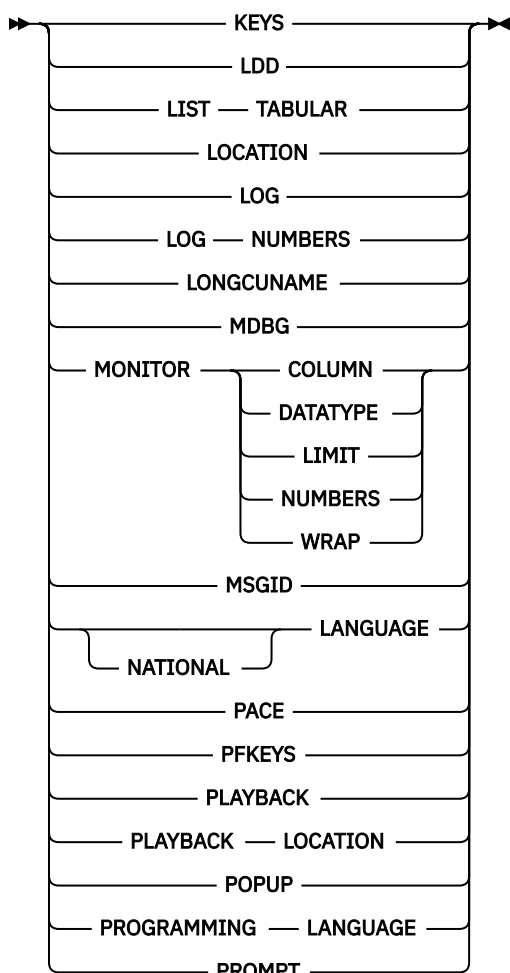
The QUERY command displays the current value of the specified z/OS Debugger setting, the current setting of all the z/OS Debugger settings, or the current location in the suspended program.


**QUERY** — **Attributes A through I** — **Attributes J through P** — **Attributes Q through Z** — ; —

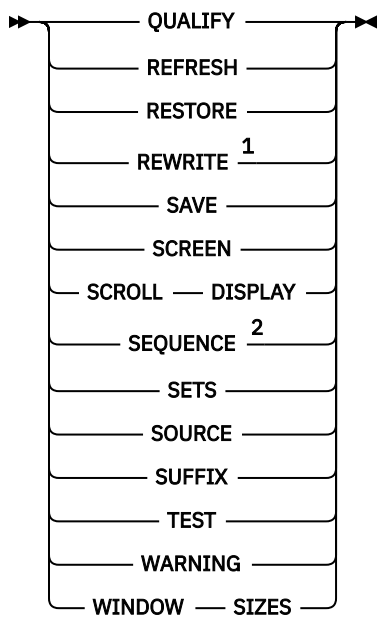
**Attributes A through I**

ASSEMBLER
AUTOMONITOR
BROWSE MODE 1
CHANGE
COLORS
COUNTRY
CURRENT — VIEW 1
DBCS
DEFAULT — DBG 1
DEFAULT — LISTINGS 1
DEFAULT — MDBG 1
DEFAULT — SCROLL
DEFAULT — VIEW 1
DEFAULT — WINDOW
DISASSEMBLY
DYNDEBUG
ECHO
EQAOPTS 1
EQUATES
EXECUTE
EXPLICITDEBUG 1
FIND BOUNDS
FREQUENCY
HISTORY
IGNORELINK 1
INTERCEPT 1

#### Attributes J through P



### Attributes Q through Z



Notes:

<sup>1</sup> You can use this command in remote debug mode.

<sup>2</sup> 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 z/OS Debugger session and if an expression is specified, sets the return code.

➤ QUIT — ( — *expression* — ) — ABEND — DEBUG — TASK — ; ➤

## QQUIT command

---

The QQUIT command ends a z/OS Debugger 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 z/OS Debugger terminated.

➤ RESTORE — SETTINGS — BPS — MONITORS — BPS — MONITORS — MONITORS — BPS — ; ➤

## 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.

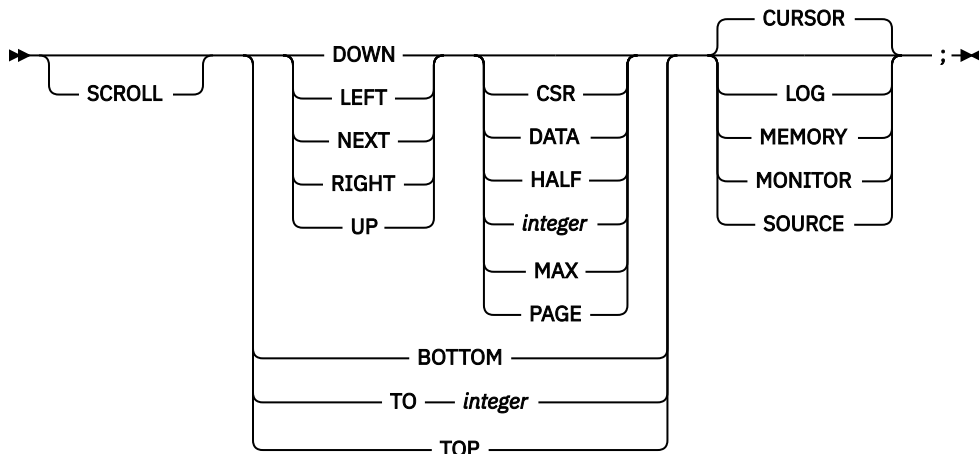
➤ RUNTO — statement\_id ; ➤

## RUNTO prefix command (full-screen mode)

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

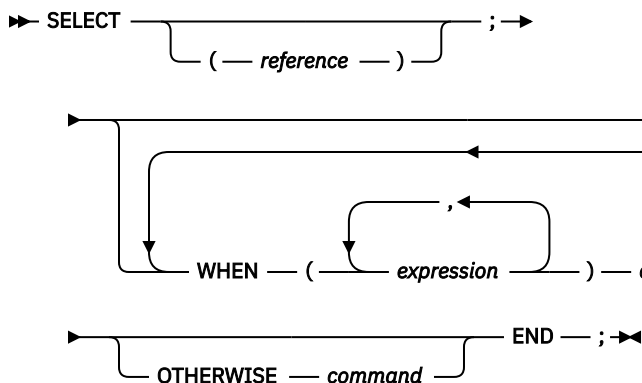
## SCROLL command (full-screen mode)

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



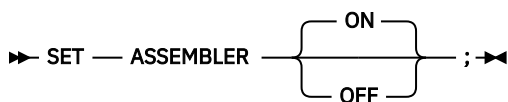
## SELECT command (PL/I)

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



## SET ASSEMBLER ON/OFF

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



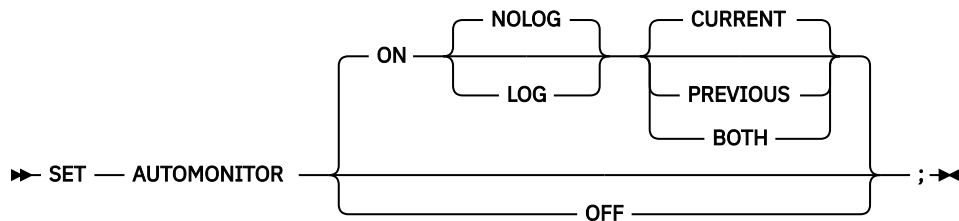
## SET ASSEMBLER STEPOVER

The SET ASSEMBLER STEPOVER command specifies how z/OS Debugger processes STEP OVER commands in assembler compile units.



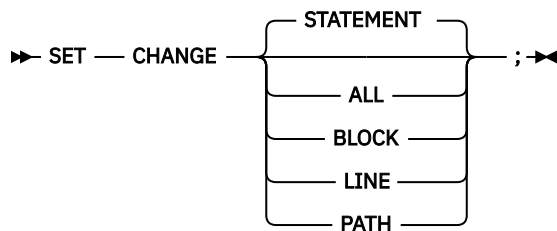
## SET AUTOMONITOR

Controls the monitoring of data items at the statement that z/OS Debugger runs next, ran most recently, or both.



## SET CHANGE

Controls the frequency of checking the AT CHANGE breakpoints.

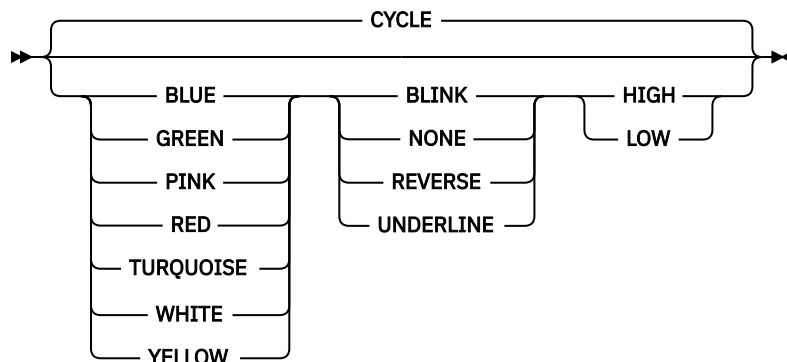


## SET COLOR (full-screen and line mode)

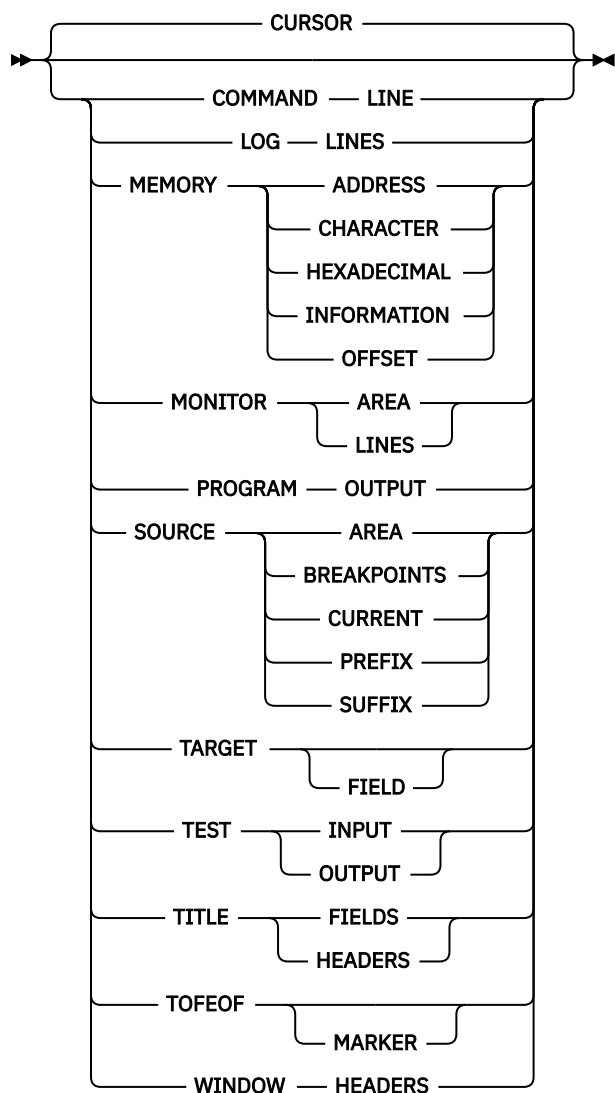
Provides control of the color, highlighting, and intensity attributes when the SCREEN setting is ON.



**color\_attributes**



**UI\_elements**



## SET COUNTRY

Changes the current national country setting for the application program.

➤ SET — COUNTRY — *country\_code* — ; ➤

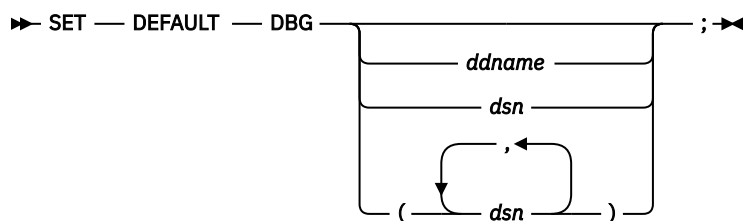
## SET DBCS

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

➤ SET — DBCS — { ON  
OFF } — ; ➤

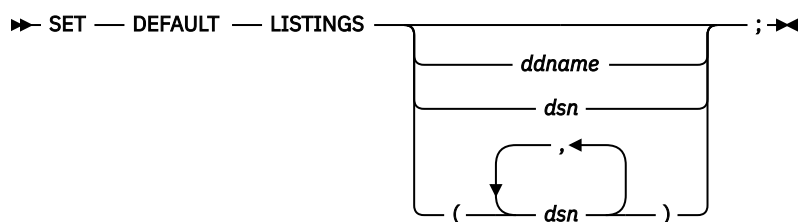
## SET DEFAULT DBG

Defines a default partitioned data set DD name or DS name that z/OS Debugger searches through to locate the .dbg files.



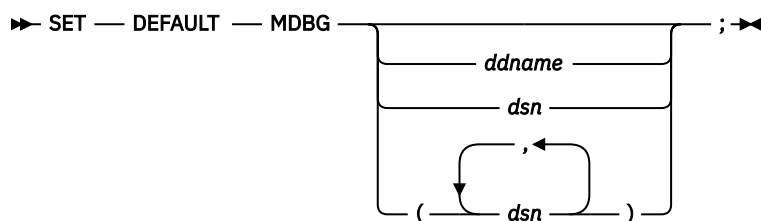
## SET DEFAULT LISTINGS

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



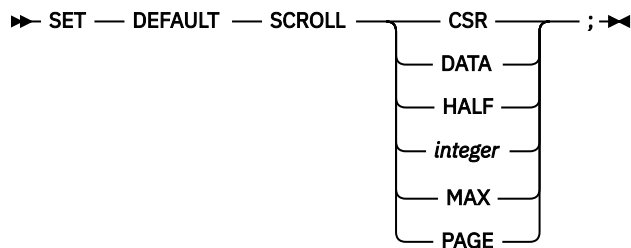
## SET DEFAULT MDBG

Defines a default partitioned data set DD name or DS name that z/OS Debugger searches through to locate the .mdbg files.



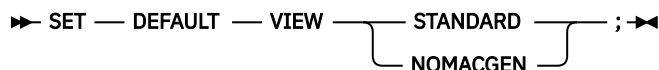
## SET DEFAULT SCROLL (full-screen mode)

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



## SET DEFAULT VIEW

Controls the default view for assembler compile units.



## 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 } { \* | keyword } ; ➤

## 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 } ; ➤

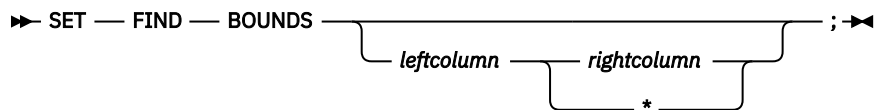
## SET EXPLICITDEBUG

Controls whether explicit debug mode is active.



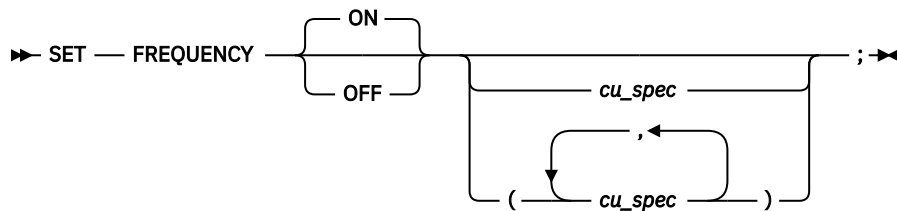
## SET FIND BOUNDS

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.



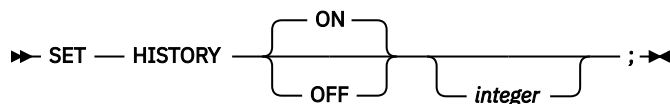
## SET FREQUENCY

Controls whether statement executions are counted.



## SET HISTORY

Specifies whether entries to z/OS Debugger are recorded in the history table and optionally adjusts the size of the table.



## SET IGNORELINK

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



## SET INTERCEPT (C and C++)

Intercepts input to and output from specified files.

```

  >> SET — INTERCEPT — { ON } — FILE — file_spec — ; >>
                             { OFF }

```

## SET INTERCEPT (COBOL, full-screen mode, line mode, batch mode)

Intercepts input to and output from the console.

```

  >> SET — INTERCEPT — { ON } — CONSOLE — ; >>
                             { OFF }

```

## SET INTERCEPT (COBOL, remote debug mode)

Intercepts output from COBOL DISPLAY statements.

```

  >> SET — INTERCEPT — { ON } — ; >>
                             { OFF }

```

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

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

```

  >> SET — KEYS — { ON } — { 12 } — ; >>
                   { OFF }  { 24 }

```

## SET LDD

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

```

  >> SET — LDD — { SINGLE } — ; >>
                  { ALL }

```

## SET LIST BY SUBSCRIPT command (COBOL)

Controls whether z/OS Debugger displays elements in an array as they are stored in memory.

```

  >> SET — LIST — BY — SUBSCRIPT — { ON } — ; >>
                                      { OFF }

```

## SET LIST BY SUBSCRIPT command (Enterprise PL/I, full-screen mode only)

Controls whether z/OS Debugger displays elements in an array as they are stored in memory.



➤ SET — LIST — BY — SUBSCRIPT — { ON / OFF } ; ➤

## SET LIST TABULAR

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

➤ SET — LIST — TABULAR — { OFF / ON } ; ➤

## SET LOG

Controls whether each command that z/OS Debugger 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 — { ON / OFF } ; ➤

➤ SET — LOG — ON — FILE — *fileid* — { OLD / MOD } ; ➤

➤ SET — LOG — OFF — KEEP — *count* ; ➤

## SET LOG NUMBERS (full-screen and line mode)

Controls whether line numbers are shown in the log window.

➤ SET — LOG — NUMBERS — { ON / OFF } ; ➤

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

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

➤ SET — LONGCUNAME — { ON / OFF } ; ➤

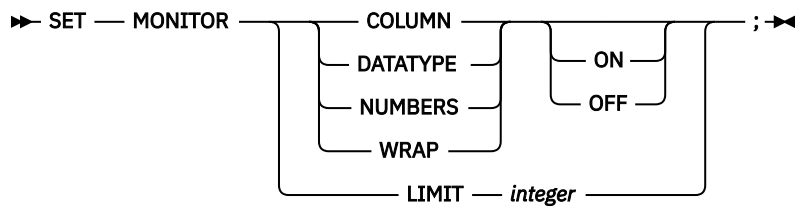
## SET MDBG (C, C++)

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

➤ SET — MDBG — (*lm\_spec*) — *fileid* — ; ➤

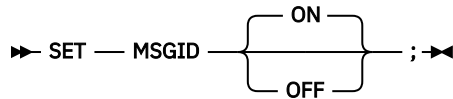
## SET MONITOR (full-screen and line mode)

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



## SET MSGID

Controls whether the z/OS Debugger messages are displayed with the message prefix identifiers.



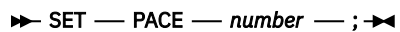
## SET NATIONAL LANGUAGE

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



## SET PACE

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



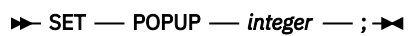
## SET PFKEY

Associates a z/OS Debugger command with a Program Function key (PF key).



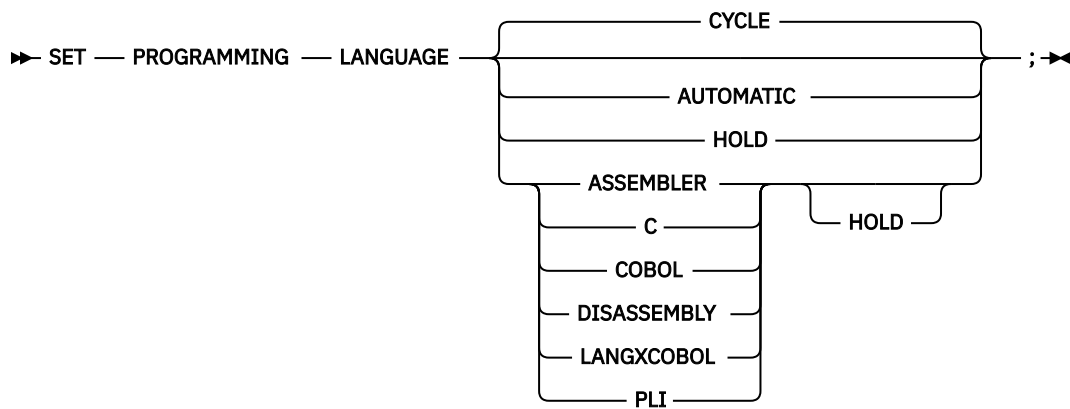
## SET POPUP

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



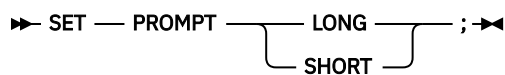
## SET PROGRAMMING LANGUAGE

Sets the current programming language.



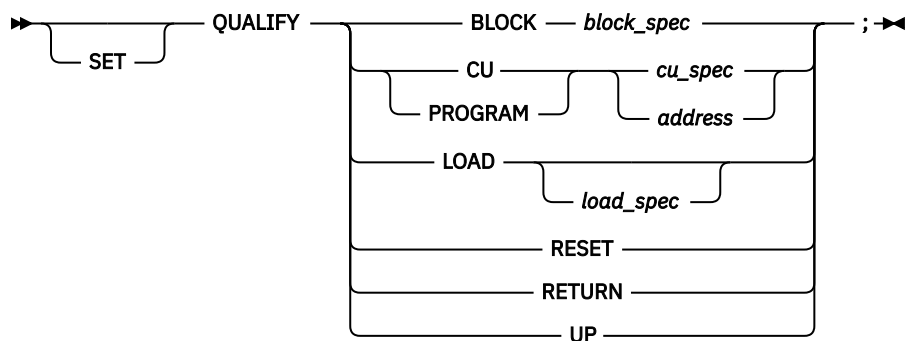
## SET PROMPT (full-screen and line mode)

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



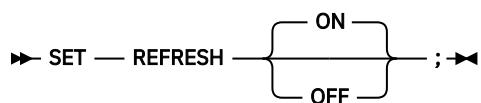
## SET QUALIFY

Simplifies the identification of references and statement numbers by resetting the point of view to a new block, compile unit, or load module.



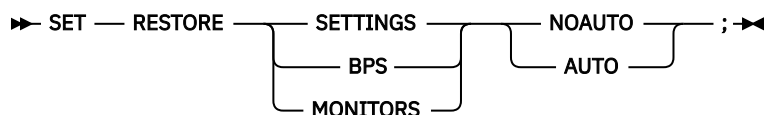
## SET REFRESH (full-screen mode)

Controls screen refreshing.



## SET RESTORE

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



## SET REWRITE (full-screen mode)

Forces a periodic screen rewrite during long sequences of output.



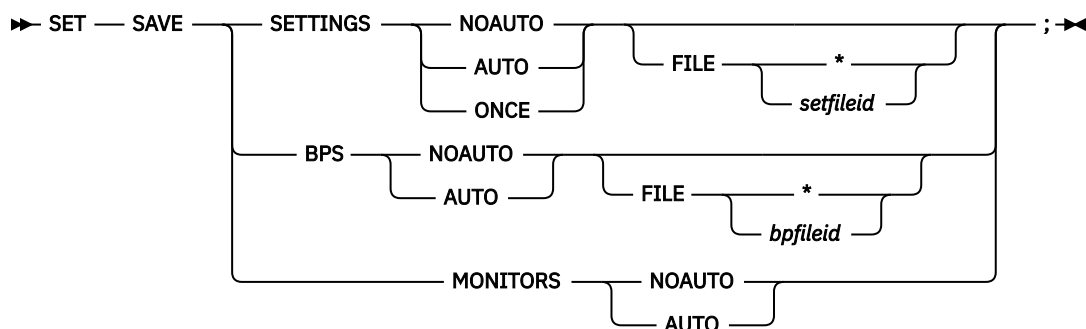
## SET REWRITE (remote debug mode)

Sets the maximum number of COBOL DISPLAY statements that the remote debugger displays in the Debug Console.



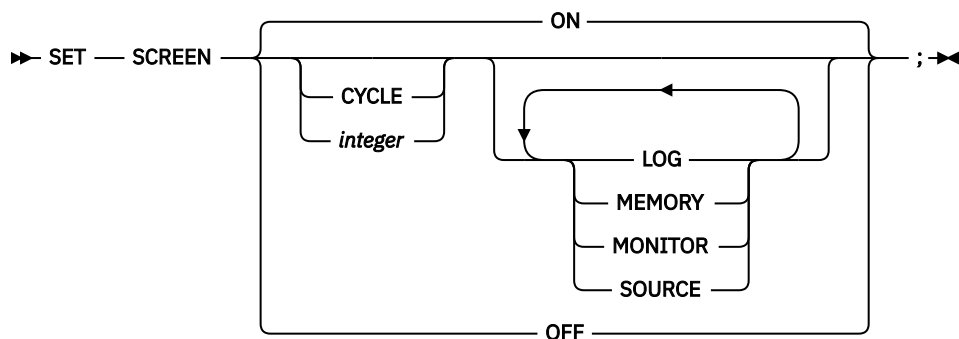
## SET SAVE

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.



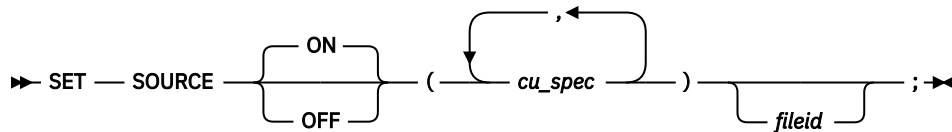
## SET SEQUENCE (PL/I)

Controls whether z/OS Debugger interprets data after column 72 in a commands or preference file as a sequence number.



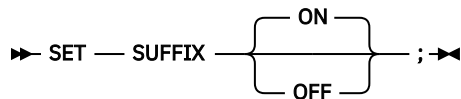
## SET SOURCE

Associates a source file, compiler listing or separate debug file with one or more compile units.



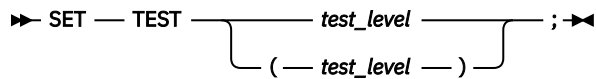
## SET SUFFIX (full-screen mode)

Controls the display of frequency counts at the right edge of the Source window when in full-screen mode.



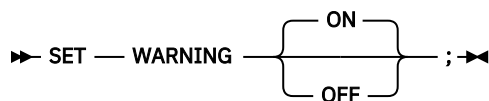
## SET TEST

Overrides the initial TEST runtime options specified at invocation.



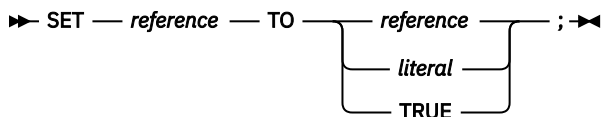
## SET WARNING (C, C++, and PL/I)

Controls display of the z/OS Debugger warning messages and whether exceptions are reflected to the application program.



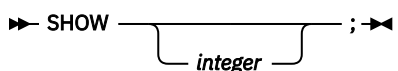
## SET command (COBOL)

The SET command assigns a value to a COBOL reference.



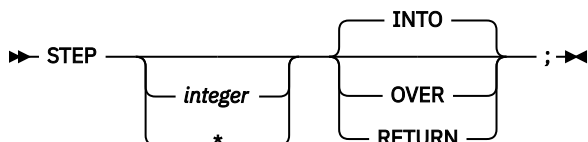
## SHOW prefix command (full-screen mode)

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.



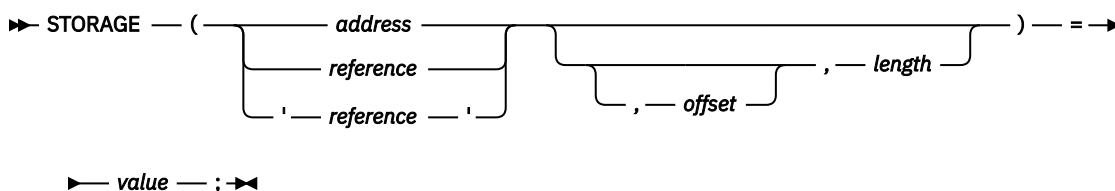
## STEP command

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



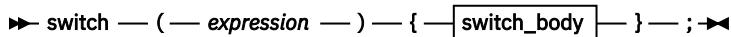
## STORAGE command

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

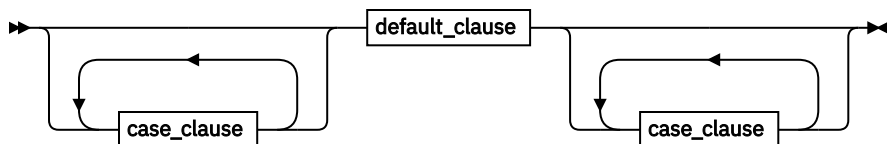


## switch command (C and C++)

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

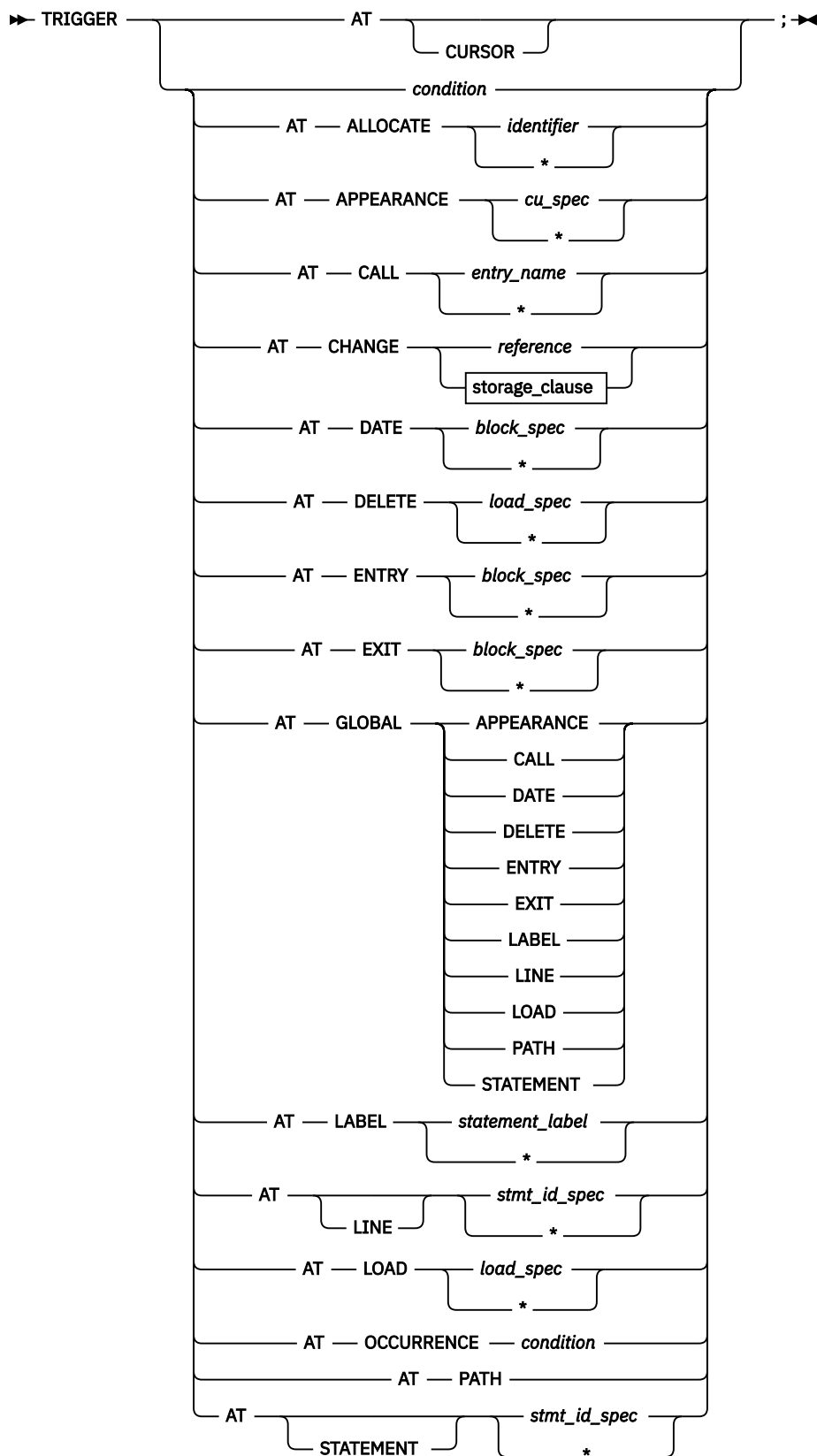


**switch\_body**



**case\_clause**





#### storage\_clause

TRIGGER %STORAGE — ( — address — , — length — ) —



## TSO command (z/OS)

The TSO command lets you issue TSO commands during a z/OS Debugger session and is valid only in a TSO environment.

➤ TSO — *tso\_command* — ; ➤

## USE command

The USE command causes the z/OS Debugger commands in the specified file or data set to be either performed or syntax checked.

➤ 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.

➤ while — ( — *expression* — ) — *command* — ; ➤

## WINDOW CLOSE

Closes the specified window in the z/OS Debugger full-screen session panel.

➤ — *WINDOW* — CLOSE — *CURSOR* — ; ➤  
                                  — LOG —  
                                  — MEMORY —  
                                  — MONITOR —  
                                  — SOURCE —

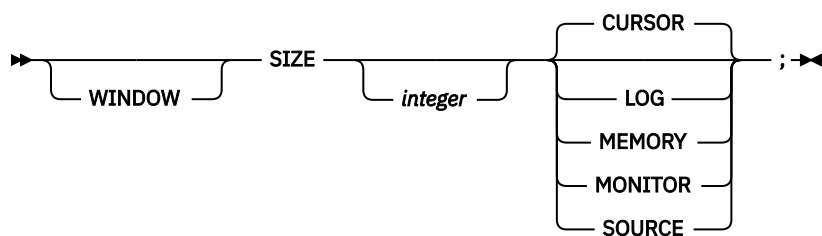
## WINDOW OPEN

Opens a previously-closed window in the z/OS Debugger full-screen session panel.

➤ — *WINDOW* — OPEN — *LOG* — ; ➤  
                                  — MEMORY —  
                                  — MONITOR —  
                                  — SOURCE —

## WINDOW SIZE

Controls the relative size of currently visible windows in the z/OS Debugger full-screen session panel.



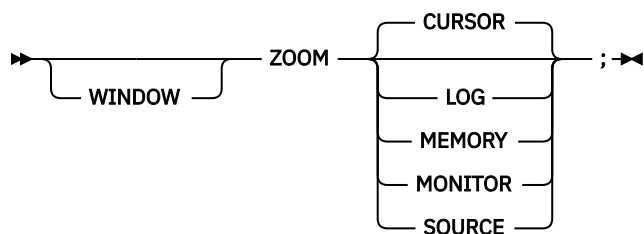
## WINDOW SWAP

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. z/OS Debugger built-in functions

z/OS Debugger provides you with the following built-in functions:

---

### **%CHAR (assembler, disassembly, and LangX COBOL)**

Returns the EBCDIC character value of an operand.

---

### **%DEC (assembler, disassembly, and LangX 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 LangX COBOL)**

Returns a string that is the address of the operand. %WHERE can be used *only* as the outermost expression in the LIST command.



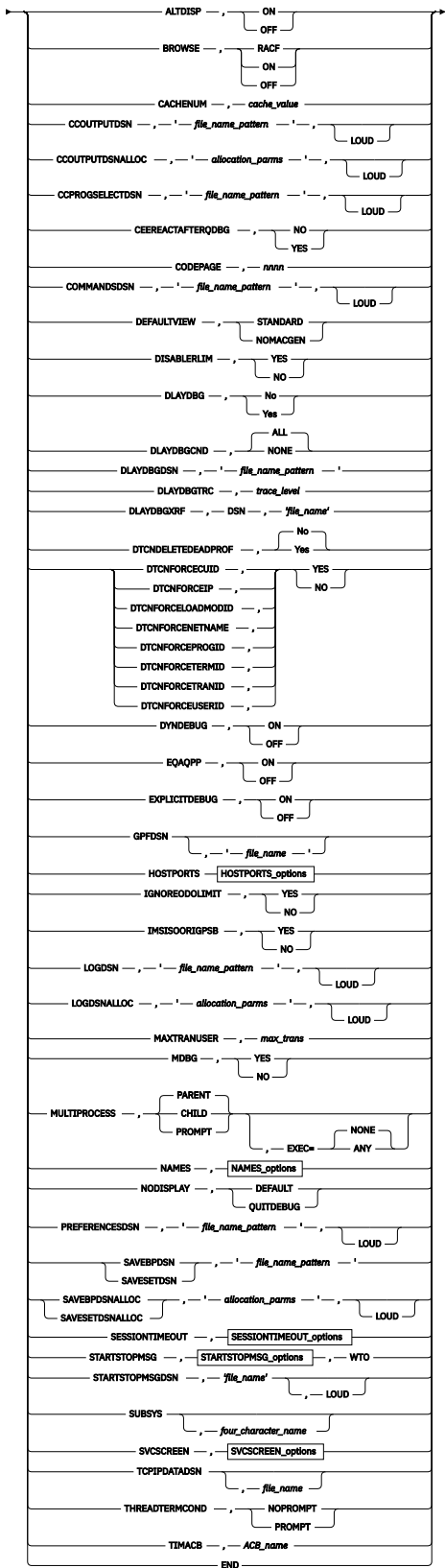
---

## Chapter 3. EQAOPTS commands (optional)

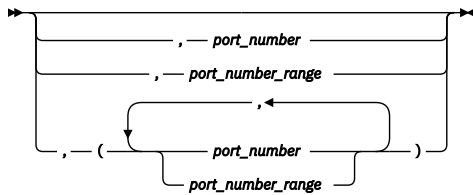
The EQAOPTS commands (formerly known as options) alter certain z/OS Debugger behaviors. This topic summarizes the syntax of the EQAOPTS commands, followed by a list briefly describing each command. For a complete description of each command and to learn how to use these commands, see *IBM z/OS Debugger Customization Guide* and *IBM z/OS Debugger Reference and Messages*.

The follow diagram describes the syntax of the EQAOPTS commands:

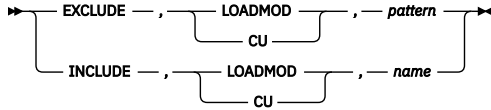
EQAXOPT →



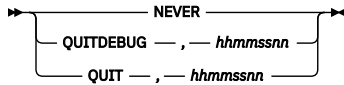
## HOSTPORTS\_options



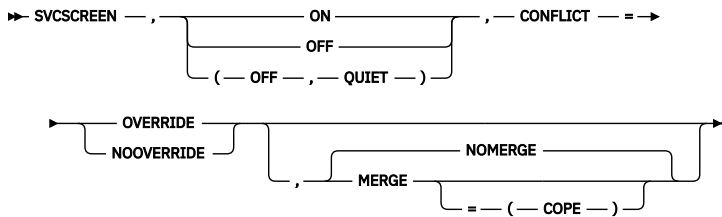
### NAMES\_options



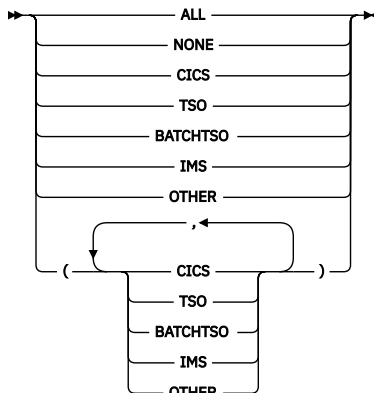
### SESSIONTIMEOUT\_options



### SVCSCREEN\_options



### STARTSTOPMSG\_options



## ALTDISP

Specifies whether to add a character indicator to indicate a breakpoint, the current line, or the line with found text. This command is valid only when you are using interactive MFI mode.

## BROWSE

Specifies whether a user with sufficient RACF® authority to the applicable browse mode RACF facility can start z/OS Debugger in browse mode.

## 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.

## CCOUTPUTDSN

---

Provides the data set name to be used for the Code Coverage Observation file. Specify NULLFILE if no Observation file is to be written to. You can also choose whether to display WTO messages that helps you debug processing done by this command.

## CCOUTPUTDSNALLOC

---

Creates the CCOUTPUTDSN data set for a new user and provides the allocation parameters (in BPXWDYN format). You can also choose whether to display WTO messages that helps you debug processing done by this command.

## CCPROGSELECTDSN

---

Provides the data set name that contains the Code Coverage Options file (which specifies the Group IDs and the PROGRAM IDs of the COBOL routines that are to be processed). Specify NULLFILE if no Code Coverage Options file is to be read. You can also choose whether to display WTO messages that helps you debug processing done by this command.

## CEEREACTAFTERQDBG

---

Specifies whether to restart z/OS Debugger with CEETEST after you use QUIT DEBUG to end a debug session.

## CODEPAGE

---

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

## COMMANDSDSN

---

Indicates the naming pattern z/OS Debugger uses to locate the data set containing a commands file. z/OS Debugger reads a member from this commands file every time it starts.

## DEFAULTVIEW

---

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

## DISABLERLIM

---

Controls whether z/OS Debugger disables Omegamon Resource Limiting (RLIM) during debug sessions in a CICS region.

## DLAYDBG

---

Enables z/OS Debugger to delay the starting of a debug session until z/OS Debugger recognizes a certain program (CU).

## DLAYDBGEND

Indicates whether you want z/OS Debugger to monitor condition events in the delay debug mode.



## DLAYDBGDSN

Indicates that z/OS Debugger is to use the specified naming pattern when it constructs the delay debug profile data set name.

## DLAYDBGTRC

Indicates that z/OS Debugger is to generate a trace during the pattern match process in the delay debug mode. z/OS Debugger uses the WTO (write to operator) command to output the trace.

## DLAYDBGXRF

Indicates that you want z/OS Debugger to use the cross reference file to find the user ID when it constructs the delay debug profile data set name.

## DTCNDELETEDEADPROF

Controls the deletion of DTCN profiles. A dead profile is a profile whose owner has logged off or disconnected from the CICS region.

## DTCNFORCExxxx

Controls DTCN behavior for the conditions described in Table 4 on page 63. When you set a DTCNFORCExxxx option to YES, DTCN forces users to specify the respective resource type. The default setting is NO.

Table 4. List of EQAOPTS DTCNFORCExxxx commands and their corresponding DTCN fields	
EQAOPTS DTCNFORCExxxx command	DTCN field name
DTCNFORCECUID or DTCNFORCEPROGID	CU
DTCNFORCEIP	IP Name/Address
DTCNFORCELOADMODID	LoadMod
DTCNFORCENETNAME	NetName
DTCNFORCETERMID	Terminal Id
DTCNFORCETRANID	Transaction Id
DTCNFORCEUSERID	User Id

### Related tasks

"Requiring users to specify resource types" in the *IBM z/OS Debugger Customization Guide*

## DYNDEBUG

Controls the initial default for the SET DYNDEBUG command.

## EQAQPP

Indicates the presence of Q++ programs.

## EXPLICITDEBUG

Controls whether explicit debug mode is active.

## GPFDN

---

Specifies the data set name for the global preferences file.

## HOSTPORTS

---

Specifies a host port or range of ports to use for a TCP/IP connection from the host to a workstation when using remote debug mode.

## IGNOREODOLIMIT

---

Instructs z/OS Debugger to display COBOL table data items even when an ODO value is out of range.

## IMSIISORIGPSB

---

Instructs the IMS Transaction Isolation Facility to preserve the original PSB of the transaction when a message is routed to a private message processing region. This command is deprecated. It is accepted for compatibility but has no effect. The original PSB is always preserved.

## LOGDSN

---

Indicates the naming pattern z/OS Debugger uses to locate the data set containing a log file.

## LOGDSNALLOC

---

Indicates the allocation parameters z/OS Debugger uses to create the data set named by the LOGDSN command.

## MAXTRANUSER

---

Defines the maximum number of IMS transactions that a single user can register using the IBM Transaction Isolation Facility.

## MDBG

---

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

## MULTIPROCESS

---

Controls the behavior of z/OS Debugger when a new POSIX process is created by a `fork()` or `exec()` function in the application.

It instructs z/OS Debugger to perform any of the following tasks when a new POSIX process is created:

- Continue debugging the current process. The current process is also referred to as the PARENT process.
- Stop debugging the current process and start debugging the newly created process. The newly created process is also referred to as the CHILD process.
- Prompt you to decide whether to follow the PARENT or CHILD process.

## 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.

## NODISPLAY

---

Controls z/OS Debugger behavior when the terminal using full-screen mode using the Terminal Interface Manager or the remote debugger are not available.

## PREFERENCESDSN

---

Indicates the naming pattern z/OS Debugger uses to locate the data set containing a preferences file. z/OS Debugger reads this preferences file every time it starts.

## 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.

## SAVEBPDSNALLOC and SAVESETDSNALLOC

---

Indicates the allocation parameters z/OS Debugger uses to create the data sets named by the SAVEBPDSN and SAVESETDSN commands.

## SESSIONTIMEOUT

---

Establishes a timeout for idle z/OS Debugger sessions

## STARTSTOPMSG

---

Controls whether to issue a message when each debugging session is initiated or terminated.

## STARTSTOPMSGDSN

Indicates that you want z/OS Debugger to write a message in the message file when the debug session is started or stopped.

## 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 z/OS Debugger allocates the source file.

## SVCSCREEN

---

Controls z/OS Debugger's enablement of SVC screening.

## TCPIPDATA DSN

---

Instructs z/OS Debugger to dynamically allocate the specified *file-name* to DDNAME SYSTCPD (if SYSTCPD is not already allocated).

## THREADTERMCOND

---

Specifies whether z/OS Debugger should suppress the prompt it displays when the thread termination condition, FINISH condition, or CEE067 is raised by Language Environment.

## 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 the *IBM z/OS Debugger Customization Guide*

## END

---

Identifies the last EQAOPTS command. You must always specify this command and make it the last command in the input stream.

## Chapter 4. z/OS Debugger variables

The following table summarizes the z/OS Debugger variables. For more information about these variables, see *IBM z/OS Debugger Reference and Messages*.

z/OS Debugger variable	Value
%ADDRESS	Address of the location where your program was interrupted
%AMODE	Current <sup>®</sup> AMODE of the suspended program
%BLOCK	Name of the current block
%CAAADDRESS	Address of the CAA control block associated with the suspended program
%CC	(Assembler and disassembly only) Condition code from current PSW
%CONDITION	Name or number of the condition when z/OS Debugger is entered because of an AT OCCURRENCE
%COUNTRY	Current country code
%CU	Name of the primary entry point of the current compile unit
%EPA	Address of the primary entry point in the current compile unit
%EPAR <sub>n</sub> or %EPRH <sub>n</sub>	(Assembler, disassembly, C and C++, and PL/I only) Extended-precision floating-point registers
%EPRB <sub>n</sub>	(Assembler and disassembly only) Extended-precision floating-point registers in binary format
%EPRD <sub>n</sub>	(Assembler and Disassembly only) Extended-precision floating-point registers in decimal format
%FPR <sub>n</sub> or %FPRH <sub>n</sub>	Single-precision floating-point registers in hexadecimal format
%FPRB <sub>n</sub>	(Assembler and Disassembly only) Single-precision floating-point registers in binary format
%FPRD <sub>n</sub>	(Assembler and Disassembly only) Single-precision floating-point registers in decimal format
%GPR <sub>n</sub>	32-bit base General Purpose Registers at the point of interruption in a program
%GPRG <sub>n</sub>	64-bit General Purpose Registers at the point of interruption in a program
%GPRH <sub>n</sub>	32-bit high General Purpose Registers at the point of interruption in a program
%HARDWARE	Type of hardware where the application is running
%LINE or %STATEMENT	Current source line number
%LOAD	Name of the load module of the current program, or an asterisk (*)
%LPR <sub>n</sub> or %LPRH <sub>n</sub>	Double-precision floating-point registers in hexadecimal format
%LPRB <sub>n</sub>	(Assembler and Disassembly only) Double-precision floating-point registers in binary format

<b>z/OS Debugger variable</b>	<b>Value</b>
%LPRD <i>n</i>	(Assembler and Disassembly only) Double-precision floating-point registers in decimal format
%NLANGUAGE	National language currently in use
%PATHCODE	Integer identifying the type of change occurring when the program flow reaches a point of discontinuity, and the path condition is raised
%PLANGUAGE	Current programming language
%PROGMASK	(Assembler and disassembly only) Program mask from current PSW
%PROGRAM	Equivalent to %CU
%PSW	(Assembler and disassembly only) Current Program Status Word
%RC	Return code from the most recent z/OS Debugger command
%RSTDSETS	A value of 1 if user settings have been restored and 0 otherwise
%RUNMODE	String identifying the presentation mode of z/OS Debugger
%R <i>n</i>	32-bit base General Purpose Registers for the currently qualified assembler or disassembly CU
%STATEMENT	Equivalent to %LINE
%SUBSYSTEM	Name of the underlying subsystem, if any, where the program is running
%SYSTEM	Name of the operating system supporting the program

---

## Chapter 5. Reference card: Frequently used z/OS Debugger commands

The following reference card provides a list of frequently used z/OS Debugger commands.

The\ reference cards are designed to be printed from a PDF file. If you are viewing this page through [IBM Documentation](#), follow these instructions to view the reference card through a PDF file and print it:

1. Click on the "PDF version" topic underneath "IBM z/OS Debugger Reference Summary" in the navigation pane. The PDF file for *IBM z/OS Debugger Reference Summary* opens.
2. In the Bookmarks (Table of Contents) view of Adobe Reader, click on the appendix heading for the reference card you want to print.
3. Scroll to the first page of the reference card, identify the page number in the Adobe Reader toolbar, then make a note of this page number. Do not use the page number printed on the bottom of the page. Scroll through the reference card to determine the number of pages it spans.
4. Click on Print in the Adobe Reader toolbar or click on **File > Print**. In the "Print Range" box, select Pages and then enter the page number you noted in the previous step, followed by a dash and the last page of the reference card. For example, if the first page of the reference card is on page 75 and the reference card spans four pages, enter "75-79" in the Pages field. If you are capable of printing in duplex, enable duplex printing. Do not alter any other setting; print these pages as you would any portrait-oriented page.
5. Click on OK or Print to start printing.

<p><b>Setting breakpoints (stopping points) at statements in a program</b></p>	<p><b>Setting breakpoints (stopping points) at the entrance or exit of a program</b></p>	<p><b>Commands that work on all breakpoints</b></p>
<p><b>A</b> A is the abbreviation for AT. Enter through the prefix area of the Source window. Sets a breakpoint on line where A is entered.</p> <p><b>PF6</b> Sets a breakpoint on line where cursor is located.</p> <p><b>AT 509</b> Sets a breakpoint on line 509.</p> <p><b>AT LABEL label_name</b> Sets a breakpoint on a label, paragraph, or section name.</p>	<p><b>AT ENTRY cu_name</b> Sets a breakpoint that stops the program when it enters cu_name.</p> <p><b>AT ENTRY *</b> Sets breakpoints that stop a program whenever z/OS Debugger enters a known program.</p> <p><b>AT EXIT cu_name</b> Sets a breakpoint that stops the program when it exits cu_name.</p> <p><b>AT EXIT *</b> Sets breakpoints that stop a program whenever z/OS Debugger exits a known program.</p>	<p><b>LIST AT</b> Displays all breakpoints in the Log window.</p> <p><b>CLEAR AT</b> Clears all breakpoints.</p> <p><b>DISABLE AT</b> Temporarily disables (deactivates) all breakpoints.</p> <p><b>ENABLE AT</b> Enables (activates) all disabled breakpoints.</p>
<p><b>Clearing (removing) breakpoints set at statements in a program</b></p> <p><b>C</b> C is the abbreviation for CLEAR. AT_Type a C in the prefix area of the Source window. When you press Enter, z/OS Debugger removes the breakpoint on the line where C is in the prefix area.</p> <p><b>PF6</b> Removes a breakpoint on line where cursor is located.</p> <p><b>CLEAR AT 509</b> Removes the breakpoint on line 509.</p> <p><b>CLEAR AT LABEL label_name</b> Clears a breakpoint from a label, paragraph, or section name.</p>	<p><b>Clearing (removing) breakpoints set at the entrance or exit of a program</b></p> <p><b>CLEAR AT ENTRY cu_name</b> Clears the breakpoint that stops the program when it enters cu_name.</p> <p><b>CLEAR AT ENTRY *</b> Clears all breakpoints that stop a program whenever z/OS Debugger enters a known program.</p> <p><b>CLEAR AT EXIT cu_name</b> Clears a breakpoint that stops the program when it exits cu_name.</p> <p><b>CLEAR AT EXIT *</b> Removes the breakpoints at every exit point in every program.</p>	<p><b>Identifying and loading a program's source and debug information</b></p> <p><b>SET DEFAULT LISTINGS source.info.library</b> Identifies a source library (PDS or PDSE) where z/OS Debugger searches for source files and debug information files. For example, SYSDEBUG files, LANGX files, and compiler listings. z/OS Debugger displays this information in the Source window.</p> <p><b>SET DEFAULT LISTINGS (source.info.lib1, source.info.lib2, ...)</b> Identifies a concatenation of source libraries (PDS or PDSE) where z/OS Debugger searches for source files and debug information files.</p> <p><b>LISTING or LIST</b> Displays a list of programs known to z/OS Debugger. Then, you can specify the name of the source file or debug information file for each program.</p> <p><b>LDD assembler_CSECT or LDD LangX_COBOL_program</b> Load debug information about assembler_CSECT or LangX_COBOL_program from the EQALANGX file into the Source window.</p>
<p><b>Setting breakpoints (stopping points) triggered by a change in the value of a variable</b></p> <p><b>AT CHANGE ITEMNO</b> Sets a breakpoint that stops the program when the value of ITEMNO changes.</p> <p><b>Clearing (removing) breakpoints triggered by a change in the value of a variable</b></p> <p><b>CLEAR AT CHANGE ITEMNO</b> Removes the breakpoint that stops the program when the value of ITEMNO changes.</p>	<p><b>Making breakpoints conditional</b></p> <p>Add a WHEN clause to make breakpoints conditional.</p> <p><b>AT CHANGE ITEMNO WHEN ITEMNO = '0805'</b> Stop after the value of ITEMNO changes, but only if ITEMNO is equal to the specified value.</p> <p><b>AT CHANGE CUSTID WHEN ACCT-BAL &gt; 100</b> Stop after the value of CUSTID changes, but only if ACCT-BAL is greater than the specified value.</p> <p><b>AT 509 WHEN ITEMNO = '0805'</b> Stop at statement 509, but only if ITEMNO is equal to the specified value.</p>	



<p><b>Displaying variables in the Monitor window</b></p> <p><b>SET AUTOMONITOR ON</b> Automatically displays the values of variables referenced by the current statement in the Monitor window.</p> <p><b>SET AUTOMONITOR ON BOTH</b> Automatically displays the values of variables referenced by both the current statement and the previously run statement in the Monitor window.</p> <p><b>MONITOR LIST ITEMNO</b> Adds the ITEMNO variable and its value to the Monitor window.</p> <p><b>SET MONITOR DATATYPE ON</b> Display the data types of variables.</p> <p><b>SET MON WRAP OFF</b> Displays values on a single line. If the value is longer than the visible area, z/OS Debugger displays a scale to indicate that there is more to see.</p> <p><b>CLEAR MONITOR</b> Clears all items from the Monitor window.</p> <p><b>C</b> C is the abbreviation for the CLEAR MONITOR command. Type in the letter C in the prefix area of the monitor window. When you press Enter, z/OS Debugger removes the variable on the line where C is in the prefix area.</p> <p><b>Changing values of variables</b></p> <p><b>Type over value displayed in the Monitor window</b> Move cursor to value displayed in Monitor window, type in new value, then press Enter.</p> <p><b>MOVE 24 to ACCUM-X</b> For COBOL programs, replace the value of ACCUM-X with 24.</p> <p><b>ACCUMX = 24</b> For some languages, replaces the value of ACCUMX with 24.</p>	<p><b>Displaying variables in Log window and controlling Log window options</b></p> <p><b>LIST CUST-ID or LIST TITLED CUST-ID</b> Displays the value of a variable. Only some programming languages require TITLED.</p> <p><b>PF4 or LIST</b> Displays the value of a variable identified by the location of the cursor.</p> <p><b>LIST TITLED WSS or LS or FS or LOS</b> Display contents of specific SECTIONS for COBOL programs. WSS means Working-Storage Section, LS means Linkage Section, FS means File Section, and LOS means Local-Storage Section.</p> <p><b>LIST TITLED *</b> Displays the values of all variables.</p> <p><b>SET ECHO OFF</b> z/OS Debugger does not display STEP and GO commands in the Log window. However, if a log file is open, z/OS Debugger writes them to the log file.</p> <p><b>SET LOG ON FILE file_name OLD</b> Opens a log file. When z/OS Debugger opens the log file, all items it writes to the Log window are also written to the log file.</p> <p><b>Refresh the Source window to display the current statement</b></p> <p><b>QUALIFY RESET</b> Repositions source in the Source window so that z/OS Debugger displays the current program and current statement.</p> <p><b>Working with called programs</b></p> <p><b>STEP or STEP INTO</b> When the current statement is a CALL, steps into the called program.</p> <p><b>STEP OVER</b> When the current statement is a CALL, z/OS Debugger runs the called program but does not display it. z/OS Debugger stops at the statement after the call.</p> <p><b>LOAD program_name</b> Make program_name known to z/OS Debugger.</p> <p><b>QUALIFY program_name</b> Displays the program program_name in the Source window. When the program is displayed in the Source window, you can set a breakpoint or work with variables in that program.</p> <p><b>QUALIFY RESET</b> Reposition to the current program and the current line.</p>	<p><b>Controlling program execution</b></p> <p><b>STEP or PF2</b> Run one statement or line.</p> <p><b>GO or PF9</b> Run the program until z/OS Debugger encounters a breakpoint, the program finishes, or an abend occurs.</p> <p><b>RUNTO 27</b> Runs the program and then stops before it runs line 27.</p> <p><b>R</b> R is the abbreviation of RUNTO. Type in the command in the prefix area of the Source window. When you press Enter, z/OS Debugger runs the program until it reaches the line with the R in the prefix area.</p> <p><b>GO BYPASS</b> Resume running a program after encountering an abend. Enter this command immediately after an abend occurs. z/OS Debugger skips the statement that caused the abend and continues running the program from the next logical statement.</p> <p><b>Skipping (do not run) over program statements</b></p> <p><b>JUMPTO 27</b> Moves the point at which the program resumes execution to line 27, does not run any statements between the current point and line 27, and then pauses at line 27. When you enter a GO or STEP command, the program resumes running at line 27.</p> <p><b>GOTO 27</b> Moves the point at which the program resumes execution to line 27, does not run any statements between the current point and line 27, and then resumes running the program at line 27.</p>
--	---	--

<u>Commands that work in the prefix area of the Source window</u>		<u>Working with PF keys</u>	<u>Displaying help for commands</u>
<b>A</b>	A is the abbreviation for AT. Sets a breakpoint on the line.	<b>QUERY PFKEYS</b> Displays the PF key settings in the log.	<b>?</b> Displays a list of commands
<b>C</b>	C is the abbreviation for CLEAR AT. Clears the breakpoint from the line.	<b>SET KEYS ON</b> z/OS Debugger displays the PF key settings for PF keys 1-12 at the bottom of the screen.	<b>AT ?</b> Example of displaying help for the AT command. Enter all or part of a command, followed by a question mark ("?)") to display keywords that are valid at the location of the question mark.
<b>D</b>	D is the abbreviation for DISABLE AT. Disables the breakpoint on the line.	<b>SET KEYS ON 24</b> z/OS Debugger displays the PF key settings for PF keys 13-24 at the bottom of the screen.	<b>Continuing a long command</b> <b>- (dash at the end of a line)</b> To continue a long command (for example, a command that exceeds the size of the command line), type a dash at the end of a partial command and then press Enter. z/OS Debugger prompts you to enter the rest of the command.
<b>E</b>	E is the abbreviation for ENABLE AT. Enables the breakpoint on the line.	<b>SET KEYS OFF</b> z/OS Debugger removes the PF key settings from the bottom of the screen.	<b>Abbreviating commands</b> <b>(use partial keywords)</b> You can abbreviate keywords in z/OS Debugger commands to the least number of letters that make the keyword unambiguous. For example, you can abbreviate the command MONITOR LIST VARX to MON LIST VARX or MO LIS VARX.
<b>L</b>	L is the abbreviation for LIST. Displays all variables referenced by the statement in the log. This prefix command works only for programs compiled with specific compilers.	<b>SET PF16 "MON" = MONITOR LIST</b> Example of assigning a command to a PF key. In this example, you assign the MONITOR LIST command to the PF16 key. When z/OS Debugger displays PF keys 13-24 at the bottom of the screen, it shows "PF16=MON".	<b>Ending a debugging session</b> <b>QUIT</b> Ends the debugging session and prompts you to verify that you want to end the debugging session.
<b>L1, L2, L3, ...</b>	L is the abbreviation for LIST. Displays the first, second, third, and so on variable referenced by the statement in the log. This prefix command works only for programs compiled with specific compilers.	<b>Default PF key settings</b> <b>PF1 or PF13</b> ? (HELP) <b>PF2 or PF14</b> STEP <b>PF3 or PF15</b> END <b>PF4 or PF16</b> LIST <b>PF5 or PF17</b> FIND <b>PF6 or PF18</b> AT/CLEAR <b>PF7 or PF19</b> UP <b>PF8 or PF20</b> DOWN <b>PF9 or PF21</b> GO <b>PF10 or PF22</b> ZOOM <b>PF11 or PF23</b> ZOOM LOG <b>PF12 or PF24</b> RETRIEVE	<b>QUIT ABEND</b> Ends the debugging session and terminates the program with an abend at the current location.
<b>M</b>	M is the abbreviation for MONITOR LIST. Displays all variables referenced by the statement in the Monitor window. This prefix command works only for programs compiled with specific compilers.		
<b>M1, M2, M3, ...</b>	M is the abbreviation for MONITOR LIST. Displays first, second, third, and so on variable referenced by the statement in the Monitor window. This prefix command works only for programs compiled with specific compilers.		
<u>Commands that work in the prefix area of the Monitor window</u>			
<b>C</b>	C is the abbreviation for CLEAR MONITOR. Removes the variable from the Monitor window.		
<b>D</b>	D is the abbreviation for default. Displays the value of the variable in a format based on its declared data type.		
<b>H</b>	H is the abbreviation for hexadecimal. Displays the value of the variable in hexadecimal format.		

---

## Chapter 6. Reference card: Frequently used z/OS Debugger commands while debugging assembler programs

The following reference card provides a list of frequently used z/OS Debugger commands while debugging assembler programs. Enter all the commands on the command line, unless otherwise indicated.

The reference cards are designed to be printed from a PDF file. If you are viewing this page through [IBM Documentation](#), follow these instructions to view the reference card through a PDF file and print it:

1. Click on the "PDF version" topic underneath "IBM z/OS Debugger Reference Summary" in the navigation pane. The PDF file for *IBM z/OS Debugger Reference Summary* opens.
2. In the Bookmarks (Table of Contents) view of Adobe Reader, click on the appendix heading for the reference card you want to print.
3. Scroll to the first page of the reference card, identify the page number in the Adobe Reader toolbar, then make a note of this page number. Do not use the page number printed on the bottom of the page. Scroll through the reference card to determine the number of pages it spans.
4. Click on Print in the Adobe Reader toolbar or click on File->Print. In the "Print Range" box, select Pages and then enter the page number you noted in the previous step, followed by a dash and the last page of the reference card. For example, if the first page of the reference card is on page 75 and the reference card spans four pages, enter "75-79" in the Pages field. If you are capable of printing in duplex, enable duplex printing. Do not alter any other setting; print these pages as you would any portrait-oriented page.
5. Click on OK or Print to start printing.

**Setting breakpoints (stopping points) in a program**

- A** Type a C in the prefix area of the Source window on line where you want to set a breakpoint. Press Enter. z/OS Debugger sets a breakpoint on that line. (A is the abbreviation for AT.)
- PF6** Move cursor to line where you want to set the breakpoint. Press PF6. z/OS Debugger sets a breakpoint on that line.
- AT 509** z/OS Debugger sets a breakpoint on line 509.
- AT EXIT \*** z/OS Debugger sets a breakpoint at every program exit point.
- AT ENTRY \*** z/OS Debugger sets a breakpoint at every program entry point.
- AT ENTRY cuname** z/OS Debugger sets a breakpoint at entry point of *cuname*.

**Clearing (removing) breakpoints set in a program**

- C** Type a C in the prefix area of the Source window on a line containing a breakpoint you want to remove. Press Enter. z/OS Debugger removes the breakpoint on that line. (C is the abbreviation for CLEAR AT.)
- PF6** Move cursor to line where you want to remove the breakpoint. Press PF6. z/OS Debugger removes the breakpoint on that line.
- CLEAR AT 509** z/OS Debugger removes the breakpoint on line 509.
- CLEAR AT EXIT \*** z/OS Debugger removes breakpoints from every program exit point.
- CLEAR AT ENTRY \*** z/OS Debugger removes breakpoints from every program entry point.
- CLEAR AT** z/OS Debugger removes all breakpoints.

**Set a deferred breakpoint**

A deferred breakpoint is a breakpoint you set on a program that is not yet known to z/OS Debugger.

- AT ENTRY cuname**  
If the block name is the same as *cuname*, you no longer have to specify *cuname* as *cuname*:>*cuname*.

**Setting conditional breakpoints**

- AT CHANGE ITEMNO**  
After the value of ITEMNO changes, z/OS Debugger stops the program. ITEMNO must be known through DC, DS, or USING.
- AT CHANGE \_STORAGE (R6 + 1::5)**  
After the data at the address (R6+1) changes, z/OS Debugger stops the program.
- Example of how to stop when a variable reaches a specific value, display line number where program stopped, then display value of ITEMNO:
- ```
AT CHANGE ITEMNO WHEN ITEMNO = '00006' DO QUERY LOC;
LIST ('ITEMNO CHANGED HERE', ITEMNO)
END; GO; ;
```
- You can substitute ITEMNO = '00006' with ITEMNO = 6 or \_STORAGE (R6 + 1::5). The AT CHANGE, WHEN, and DO clause must be on the same line.

**Clearing (removing) conditional breakpoints**

- CLEAR AT CHANGE ITEMNO**  
z/OS Debugger removes the breakpoint that stops the program when the value of ITEMNO changes.
- Viewing data within a program**
- MONITOR LIST ITEMNO SET MON WRAP OFF**  
z/OS Debugger adds the variable ITEMNO and its current value to the Monitor window, and displays everything on one line.

**LIST ITEMNO**

z/OS Debugger adds contents of ITEMNO to the log.

**SET AUTO ON**

z/OS Debugger automatically displays values in the Monitor window.

**SET AUTO ON LOG**

z/OS Debugger automatically displays values in the log.

**Monitor window prefix commands**

Type these letters into the prefix area of the Monitor window.

- C** z/OS Debugger runs the CLEAR MONITOR *n* command, where *n* is the monitor number assigned to the variable being monitored on that line.
- H** Display the value of the variables on this line in hexadecimal format.
- DEF** Display the value of the variables on this line in their declared format.
- Changing values of variables**
- ITEMNO = '00006'**  
z/OS Debugger replaces the value of ITEMNO with 00006.

**Type over the displayed value**

In the Monitor window, if the value is between black boxes, type the new value over the displayed value.

**Working with called programs****STEP INTO**

z/OS Debugger steps into the program being called and continues stepping through that program.

**STEP OVER**

z/OS Debugger runs the program being called without displaying the source for that program.

Enter the following series of commands to do the following tasks:

1. Identify *prgname* to z/OS Debugger.
2. Be able to set a breakpoint in *prgname*.
3. Display the source of *prgname* in the Source window.

```
LOAD prgname;
QUALIFY prgname
QUALIFY RESET;
```

**Return to or find the next line to run****QUALIFY RESET**

z/OS Debugger displays, in the Source window, the next line it will run.

**Load assembler & OSVS COBOL debugging information****LDD asmsect**

z/OS Debugger loads debug information from EQALANGX into the Source window.

**LDD oscob, prog**

z/OS Debugger loads debug information from EQALANGX into the Source window.

| <u>Working with registers</u>                                                                                                                                          | <u>Memory window</u>                                                                                                                                                                             | <u>Suggested PF key settings</u>                |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------|
| <b>MON LI REG</b><br>z/OS Debugger adds all registers to the Monitor window.                                                                                           | <b>MEMORY ITEMNO</b><br>z/OS Debugger displays raw storage beginning at the location of ITEMNO.                                                                                                  | <b>PF1</b><br>"PF13/24"=SET KEYS 24;            |
| <b>LIST STORAGE (%GPR6-&gt;+1,5) ;</b><br>z/OS Debugger display 5 bytes of storage starting at the address pointed to by register 6 plus 1 byte.                       | <b>MEMORY X'2503D008'</b><br>z/OS Debugger displays memory beginning base address X'2503D008'..                                                                                                  | <b>PF2</b><br>"KEEP"=MON LOCAL %CU LIST CURSOR; |
| <b>MON LI STORAGE (R6-&gt;+1,5) ;</b><br>z/OS Debugger adds to the Monitor window the 5 bytes of storage starting at the address pointed to by register 6 plus 1 byte. | <b>SWAP MEMORY LOG</b><br>z/OS Debugger switches between the Memory window and the Log window.                                                                                                   | <b>PF3</b><br>"QUIT"=QUIT;                      |
| <b>MON LI STORAGE (R6 + 1:15)</b><br>z/OS Debugger adds to the Monitor window the 5 bytes of storage starting at the address pointed to by register 6 plus 1 byte.     | <b>SET EQUATE SW= SWAP MEM LOG'</b><br>z/OS Debugger assigns the letters "SW" with the command SWAP MEMORY LOG. Afterwards, you enter SW to switch between the Memory window and the Log window. | <b>PF4</b><br>"CURRLINE"=QUALIFY RESET          |
| <b>Controlling program execution</b>                                                                                                                                   | <b>CICS pseudo-conversational programs</b>                                                                                                                                                       | <b>PF5</b><br>"FIND"=FIND;                      |
| <b>STEP</b><br>Run one instruction.                                                                                                                                    | <b>SET TEST ERROR</b><br>z/OS Debugger does not stop for Language Environment class 1 exceptions: EXEC CICS RETURN and STOP RUN in batch.                                                        | <b>PF7</b><br>"UP"=SCROLL UP;                   |
| <b>PF2</b><br>PF key to run one instruction.                                                                                                                           | <b>QUIT DEBUG TASK</b><br>Stop z/OS Debugger for this task.                                                                                                                                      | <b>PF8</b><br>"DOWN"=SCROLL DOWN;               |
| <b>GO</b><br>Run the program until it abends, encounters a breakpoint, or finishes.                                                                                    | <b>Technique to bypass an abend condition</b>                                                                                                                                                    | <b>PF9</b><br>"GO 1"=STEP                       |
| <b>PF9</b><br>PF key to run the program until it abends, encounters a breakpoint, or finishes.                                                                         | <b>GO BYPASS</b><br>Skip the exception condition and stop at the next breakpoint.                                                                                                                | <b>PF10</b><br>"ZOOM WIN"=IMMEDIATE ZOOM;       |
| <b>JUMPTO 27</b><br>z/OS Debugger makes line 27 the current point of execution, but does not run line 27.                                                              | <b>Technique to call Fault Analyzer for DUMP</b>                                                                                                                                                 | <b>PF11</b><br>"ZOOM LOG"=WINDOW ZOOM LOG;      |
| <b>RUNTO 27</b><br>z/OS Debugger runs the program to line 27, then stops.                                                                                              | <b>CALL %FA</b><br>z/OS Debugger produces a DUMP that Fault Analyzer can use in its analysis.                                                                                                    | <b>PF12</b><br>"GO"=GO;                         |
| <b>Saving monitors, breakpoints and settings across debugging sessions</b>                                                                                             | <b>End program debugging (testing)</b>                                                                                                                                                           | <b>PF13</b><br>"PF1/12"=SET KEYS 12;            |
| <b>SAVE MONITORS</b><br>z/OS Debugger saves all variables being monitored in the Monitor window.                                                                       | <b>QUIT</b><br>Ends z/OS Debugger and program stops running.                                                                                                                                     | <b>PF14</b><br>"CLR KEEP"=CLEAR MONITOR;        |
| <b>SAVE BPS</b><br>z/OS Debugger saves all the breakpoints.                                                                                                            | <b>QQUIT</b><br>Ends z/OS Debugger without prompt.                                                                                                                                               | <b>PF15</b><br>"QUIT"=QUIT;                     |
| <b>SAVE SETTINGS</b><br>z/OS Debugger saves all the settings set by the SET command.                                                                                   | <b>QUIT DEBUG</b><br>Ends z/OS Debugger but program continues running.                                                                                                                           | <b>PF16</b><br>"LIST CSR"=LIST CURSOR;          |
| <b>RESTORE MONITORS</b><br>z/OS Debugger restores all the variables saved by the SAVE MONITORS command to the Monitor window.                                          | <b>QUIT DEBUG TASK</b><br>Ends z/OS Debugger but program continues running.                                                                                                                      | <b>PF17</b><br>"FIND"=IMMEDIATE FIND;           |
| <b>RESTORE BPS</b><br>z/OS Debugger restores all the breakpoints saved by the SAVE BPS command.                                                                        | <b>QUIT ABEND</b><br>Causes a ROLLBACK for IMS and Db2 programs.                                                                                                                                 | <b>PF18</b><br>"AT/CLEAR"=AT TOGGLE;            |
| <b>RESTORE SETTINGS</b><br>z/OS Debugger restores all the settings saved by the SAVE SETTINGS command.                                                                 |                                                                                                                                                                                                  | <b>PF19</b><br>"TOP"=TOP;                       |
|                                                                                                                                                                        |                                                                                                                                                                                                  | <b>PF20</b><br>"BOTTOM"=BOT;                    |
|                                                                                                                                                                        |                                                                                                                                                                                                  | <b>PF21</b><br>"GO 1"=STEP;                     |
|                                                                                                                                                                        |                                                                                                                                                                                                  | <b>PF22</b><br>"ZOOM WIN"=IMMEDIATE ZOOM;       |
|                                                                                                                                                                        |                                                                                                                                                                                                  | <b>PF23</b><br>"ZOOM LOG"=WINDOW ZOOM LOG;      |
|                                                                                                                                                                        |                                                                                                                                                                                                  | <b>PF24</b><br>"RETRIEVE"=IMMEDIATE RETRIEVE;   |



## 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 illustrate 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 function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (your company name) (year).

## Privacy policy considerations

---

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, or to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at <http://www.ibm.com/privacy> and IBM's Online Privacy Statement at <http://www.ibm.com/privacy/details> in the section entitled "Cookies, Web Beacons and Other Technologies", and "the IBM Software Products and Software-as-a-Service Privacy Statement" at <http://www.ibm.com/software/info/product-privacy>.

## Programming interface information

---

This document 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 z/OS Debugger.

## Trademarks and service marks

---

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered 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](http://www.ibm.com/legal/copytrade.shtml).

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







Product Number: 5724-T07